Computational Geometry 48 (2015) 147–168

Contents lists available at ScienceDirect

Computational Geometry: Theory and Applications

www.elsevier.com/locate/comgeo

A sensor-based framework for kinetic data compression ☆

Sorelle A. Friedler [a,*], David M. Mount [b,1]

[a] Dept. of Computer Science, Haverford College, Haverford, PA 19041, USA
[b] Dept. of Computer Science, University of Maryland, College Park, College Park, MD 20742, USA

ARTICLE INFO

Article history:
Received 1 December 2013
Accepted 10 September 2014
Available online 16 September 2014

Keywords:
Kinetic data
Sensor data
Lossless compression
Information theory

ABSTRACT

We introduce a framework for storing and processing kinetic data observed by sensor networks. These sensor networks generate vast quantities of data, which motivates a significant need for data compression. We are given a set of sensors, each of which continuously monitors some region of space. We are interested in the kinetic data generated by a finite set of objects moving through space, as observed by these sensors. Our model relies purely on sensor observations; it allows points to move freely and requires no advance notification of motion plans. Sensor streams are represented as random processes, where nearby sensors may be statistically dependent. We model the local nature of sensor networks by assuming that two sensor streams are statistically dependent only if the two sensors are among the m nearest neighbors of each other.

We present an algorithm for the lossless compression of the data produced by the network. We show that, under the statistical dependence and locality assumptions of our framework, asymptotically this compression algorithm encodes the data to within a constant factor of the information-theoretic lower bound dictated by the joint entropy of the system. In order to justify our locality assumptions, we provide a theoretical comparison with a variant of the kinetic data structures framework and experimental results demonstrating the existence of such locality properties in real-world data. We also give a relaxed version of our sensor stream independence property where even distant sensor streams are allowed some limited dependence. We extend the current understanding of empirical entropy to introduce definitions for joint empirical entropy, conditional empirical entropy, and empirical independence. We show that, even with the notion of limited independence and in both the statistical and empirical settings, the introduced compression algorithm achieves an encoding size that is within a constant factor of the optimal.

1. Introduction

There is a growing appreciation of the importance of algorithms and data structures for processing large data sets arising from the use of sensor networks, particularly for the statistical analysis of objects in motion. Large wireless sensor networks

☆ A preliminary version of this work titled *Compressing Kinetic Data From Sensor Networks* appeared in the *Proceedings of the 5th International Workshop on Algorithmic Aspects of Wireless Sensor Networks (AlgoSensors)*, 2009 [16].

* Corresponding author.
E-mail addresses: sorelle@cs.haverford.edu (S.A. Friedler), mount@cs.umd.edu (D.M. Mount).
URLs: http://www.cs.haverford.edu/faculty/sorelle (S.A. Friedler), http://www.cs.umd.edu/~mount (D.M. Mount).

[1] The work of David Mount has been supported by the National Science Foundation under grants CCR-0635099 and CCF-1117259 and the Office of Naval Research under grant N00014-08-1-1015.

http://dx.doi.org/10.1016/j.comgeo.2014.09.002
0925-7721/

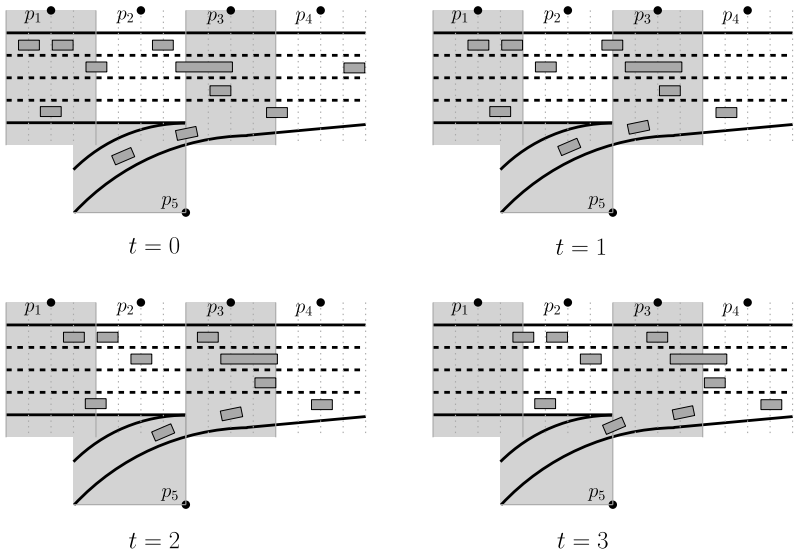

Fig. 1. An example of our sensor observation framework on a highway. Five sensors are placed at points p_1 through p_5 and observe the regions in front of them as indicated by the alternating shading patterns. Assuming that each vehicle stays in its lane and moves to the right at each time step by one unit (as marked by the light gray dotted lines), we have the following sensor counts for the four time steps shown (where the observation stream X_i is a 4-element sequence whose jth element is the number of cars that overlap p_i's region at time $t = j$): $X_1 = \langle 4, 3, 2, 1\rangle$, $X_2 = \langle 3, 2, 3, 4\rangle$, $X_3 = \langle 4, 4, 4, 4\rangle$, $X_4 = \langle 2, 1, 1, 3\rangle$, and $X_5 = \langle 2, 1, 1, 1\rangle$.

are used in areas such as road-traffic monitoring [35], environment surveillance [28], and wildlife tracking [29,39]. With the development of sensors of lower cost and higher reliability, the prevalence of applications and the need for efficient processing will increase. We are interested not in the networking aspect of these sensor systems, but rather in the fact that data is gathered from a spatially distributed set of sensors each with a limited observation region.

Before reviewing the existing literature, we give a high-level overview of our sensor-based framework for data arising from moving objects, which will be described in greater detail in Section 2. We assume we are given a fixed set of sensors, which are modeled as points in some metric space. (An approach based on metric spaces, in contrast to standard Euclidean space, offers greater flexibility in how distances are defined between objects. This is useful in wireless settings, where transmission distance may be a function of non-Euclidean considerations, such as topography and the presence of buildings and other structures.) Each sensor is associated with a region of space, which it monitors. The moving entities are modeled as points that move over time. At regular time intervals, each sensor computes statistical information about the points within its region, which are streamed as output. We refer each of these as the sensor's *observation stream*. For the purposes of this paper, we assume that this information is simply an *occupancy count* of the number of entities that overlap the sensor's region at the given time instant (for an example, see Fig. 1). Thus, we follow the minimal assumptions made by Gandhi et al. [18] and do not rely on a sensor's ability to accurately record distance, angle, etc.

Wireless sensor networks record vast amounts of data. For example, road-traffic camera systems [35] that videotape congestion produce many hours of video or gigabytes of data for analysis even if the video itself is never stored and is instead represented by its numeric content. In order to analyze trends in the data, perhaps representing the daily rush hour or weekend change in traffic patterns, many weeks or months of data from many cities may need to be stored. As the observation time or number of sensors increases, so does the total data that needs to be stored in order to perform later queries, which may not be known in advance.

In this paper we consider the problem of how to compress the massive quantities of data that are streamed from large sensor networks. Compression methods can be broadly categorized as being either *lossless* (the original data is fully recoverable), or *lossy* (information may be lost through approximation). Because lossy compression provides much higher compression rates, it is by far the more commonly studied approach in sensor networks. Our ultimate interest is in scientific applications involving the monitoring of the motion of objects in space, where the loss of any data may be harmful to the subsequent analysis. For example, in habitat monitoring [28] if the data is collected and then studied later at a time when it is not possible to re-collect earlier data, it would be important not to lose any information. In such applications it is appropriate to focus on the less studied problem of lossless compression of sensor network data. Virtually all lossless compression techniques that operate on a single stream (such as Huffman coding [24], arithmetic coding [33], Lempel–Ziv [44]) rely on the statistical redundancy present in the data stream in order to achieve high compression rates. In the context of sensor networks, this redundancy arises naturally due to correlations in the streams of sensors that are spatially close to each other. As with existing methods for lossy compression [12,19], our approach is based on aggregating correlated streams and compressing these aggregated streams.

We are particularly interested in *kinetic data*, by which we mean data arising from the observation of a discrete set of objects moving in time (as opposed to continuous phenomena such as temperature). We explore how best to store and process these assembled data sets for the purposes of efficient retrieval, visualization, and statistical analysis of the information contained within them. The data sets generated by sensor networks have a number of spatial, temporal, and statistical properties that render them interesting for study. We assume that we do not get to choose the sensor deployment based on object motion (as done in [32]), but instead use sensors at given locations to observe the motion of a discrete set of objects over some domain of interest. Thus, it is to be expected that the entities observed by one sensor will also likely be observed by nearby sensors, albeit at a slightly different time. For example, many of the vehicles driving by one traffic camera are likely to be observed by nearby cameras, perhaps a short time later or earlier. If we assume that the data can be modeled by a random process, it is reasonable to expect that a high degree of statistical dependence exists between the data streams generated by nearby sensors. If so, the information content of the assembled data will be significantly smaller than the size of the raw data. In other words, the raw sensor streams, when considered in aggregate, will contain a great deal of redundancy. Well-designed storage and processing systems should capitalize on this redundancy to optimize space and processing times. In this paper we propose a statistical model of kinetic data as observed by a collection of fixed sensors. We will present a method for the lossless compression of the resulting data sets and will show that this method is within a constant factor of the asymptotically optimal bit rate, subject to the assumptions of our model.

Although we address the problem of compression here, we are more generally interested in the storage and processing of large data sets arising from sensor networks [12,13,34,21,22]. This will involve the retrieval and statistical analysis of the information contained within them. Work resulting from the initial version of this paper has considered retrieval via spatio-temporal range searching [17]. Thus, we will discuss compression within the broader context of a framework for processing large kinetic data sets arising from a collection of fixed sensors. We feel that this framework provides a useful context within which to design and analyze efficient data structures and algorithms for kinetic sensor data.

The problem of processing kinetic data has been well studied in the field of computational geometry in a standard computational setting [23,3,36,37,5,25]. A survey of practical and theoretical aspects of modeling motion can be found in [2]. Much of the existing work applies in an online context and relies on *a priori* information about point motion. The most successful of these frameworks is the *kinetic data structure* (KDS) model proposed by Basch, Guibas, and Hershberger [5]. The basic entities in this framework are points in motion, where the motion is expressed as flight plans that are polynomials of bounded degree. Geometric structures are maintained through a set of boolean conditions, called *certificates*, and a set of associated update rules. The efficiency of algorithms in this model is a function of the number of certificates involved and the efficiency of processing them.

As valuable as KDS has been for developing theoretical analyses of point motion (see [20] for a survey), it is unsuitable for many real-world contexts and for theoretical problems that do not have locally determined properties. The requirements of algebraic point motion and advance knowledge of flight plans are either inapplicable or infeasible in many scientific applications. Recently, de Berg, Roeloffzen, and Speckmann addressed some of these issues by proposing a *black-box model* in which the point's position is updated at regular time steps and there is a point displacement bound as well as a restriction on the point density, but point motion is not otherwise restricted or known in advance. Analysis is done in terms of these bounds as well as in terms of the spread of the points [11]. This model takes important steps towards a more realistic framework for moving objects. Due to our focus on data generated by sensors, we employ a different approach.

Another approach to addressing the issue of theoretical analyses within realistic contexts is that of Buchin et al. in their recent work on algorithms for movement ecology [7,6]. They consider the issue of animal tracking data obtained with low sampling rates, for which it would be unreasonable to assume linear trajectories between samples. Instead, they use a Brownian bridge movement model between samples and consider the probability that an ecologically significant event (e.g., an encounter between two animals) has occurred at a given time.

There has also been study of algorithms that involve the distributed online processing of sensor-network data. One example is the *continuous distributed model* described by Cormode et al. [9]. This model contains a set of sensors, each of which observe a stream of data describing the recent changes in local observations. Each sensor may communicate with any other sensor or with a designated central coordinator. Efficiency is typically expressed as a trade-off between communication complexity and accuracy. This framework has been successfully applied to the maintenance of a number of statistics online [9,8,4]. Another example is the competitive online tracking algorithm of Yi and Zhang [43], in which a tracker-observer pair coordinate to monitor the motion of a moving point. Again, complexity is measured by the amount of communication between the tracker and the observer. The idea of the tracker and observer is reminiscent of an earlier model for incremental motion by Mount et al. [31]. Unlike these models, our framework applies in a traditional (non-distributed) computational setting.

The survey paper by Agarwal et al. [2] identifies fundamental directions that future research should pursue. (Even though [2] was published over a decade ago, many of the research issues identified there are still of current relevance.) Our work addresses three of these issues; unpredicted motion, motion-sensitivity, and theoretical discrete models of motion. In our framework we will process a point set without predicted knowledge and no matter its motion. *Motion-sensitive algorithms* admit complexity analyses based on the underlying motion. Imagine a set of points following a straight line or moving continuously in a circle; a well-designed algorithm calculating statistical information about such a point set should be more efficient than the same algorithm operating on a set of randomly moving points. Our motion-sensitive framework will pay a cost in efficiency based on the information content of the point motion. Finally, Agarwal et al. note that most

theoretical work relies on continuous frameworks while applied work experimentally evaluates methods based on discrete models. Our framework assumes a discrete sampling model, but is still theoretically sound. Unlike KDS, which maintains a structure through a prescribed motion sequence, we are interested in processing sensed data for the sake of subsequent analysis and retrieval.

We will establish a pure framework as well as several practical relaxations of the framework. Here, we first describe the basic assumptions of the pure framework. As mentioned above, our objective is to compress the collected sensor data in a lossless manner by exploiting redundancy in the sensor streams. In order to establish formal bounds on the quality of this compression, we assume in this pure version of the framework (as is common in entropy encoding) that the observation stream of each sensor can be modeled as a stationary, ergodic random process. We allow for statistical dependencies between the sensor streams. Shannon's source coding theorem implies that, in the limit, the minimum number of bits needed to encode the data is bounded from below by the normalized joint entropy of the resulting system of random processes [10]. There are known lossless compression algorithms, such as Lempel–Ziv [44], that achieve this lower bound asymptotically. Assuming that the number of sensors is large, it would be infeasible, however, to apply this observation *en masse* to the entire joint system of all the sensor streams. Instead, we would like to partition the streams into small subsets, and compress each subset independently. The problem in our context is how to bound the loss of efficiency due to the partitioning process. In order to overcome this problem we need to impose limits on the degree of statistical dependence among the sensors. Our approach is based on a locality assumption. Given a parameter m, we say that a sensor system is *m-local* if each sensor's stream is statistically dependent on only its m nearest sensors.

In its purest form, as described briefly above, our framework makes some assumptions that may not be satisfied in practice. In particular, it has two significant drawbacks. The first is an analysis based in the statistical setting using Shannon entropy and its extensions. These entropy definitions assume an underlying random process that generates the data, and their derived properties hold in the limit as the length of the sequence approaches infinity. When analyzing a specific data set these assumptions are too strict, since we would like these entropy properties to hold for sequences of finite length. In one of our relaxed versions of this framework, we extend the framework analysis to hold under the more realistic definition of empirical entropy [26] that has the advantage of not assuming an underlying stationary, ergodic random process. Empirical entropy relies only on the observed probabilities of the sensor data values. In order to perform the complex analyses for the framework in the empirical setting, we also introduce new definitions for empirical entropy constructs that are analogous to existing statistical ones: joint empirical entropy, conditional empirical entropy, and empirical independence.

The second modification to the basic version of the framework that should be made in order to create a more realistic analysis concerns its assumptions of independence. The pure framework makes the assumption that sensor streams are dependent only on their neighbors and are purely independent of all other streams. However, it may be the case that there is some underlying dependence that may be common to many or all sensor streams. For example, if the sensors are detecting and reporting car traffic counts (see, for example, Fig. 1), while nearby sensors may be more likely to see the same traffic patterns at consecutive time intervals, all sensors are likely to see a decrease in traffic at night and increases during rush hours. In order to analyze these underlying commonalities in the context of the framework for kinetic sensor data we introduce a notion of limited independence in both the statistical and empirical settings.

In addition to the introduction of this new framework, the main result of this paper is a lossless compression algorithm within this framework that achieves an encoding size on the order of the optimal size, even under an assumption of limited independence for both the statistical and empirical settings. The full contributions of this paper are described in the following sections. In Section 2, we introduce a new framework (in its pure form) for the compression and analysis of kinetic sensor data. In Section 3 we consider the properties of entropy and independence within statistical and empirical contexts. This examination provides the fundamental theoretical building blocks for the analyses in later sections. In Section 4 we justify and broaden the locality and statistical independence assumptions described in the basic framework introduction in Section 2. This includes a theoretical justification of our m-local model as compared to a variant of the KDS model. We prove that the compressed data from our model takes space on the order of the space used by the KDS variant. We also give experimental justification showing that the assumptions of the m-local model are borne out by real-world data. Finally, in Section 5, we prove that any m-local system that resides in a metric space of constant doubling dimension (definitions below) can be partitioned in the manner described above, so that joint compressions involve groups of at most $m+1$ sensors. We show that the final compression is within a factor c of the information-theoretic lower bound, where c is independent of m, and depends only on the dimension of the space. Additionally, we show that similar bounds hold in both statistical and empirical settings when the framework assumes limited independence between sensor streams.

2. Data framework

In this section we present a formal model of the essential features of the sensor networks to which our results will apply. Our main goal is to realistically model the data sets arising in typical wireless sensor-networks when observing kinetic data while also allowing for a clean theoretical analysis. While the framework will be modified in future sections to address potential concerns with its applicability to real-world data, here we present the framework in its purest form.

We assume a fixed set of S sensors operating over a total time period of length T. The sensors are modeled as points in some metric space. We may think of the space as Euclidean d-dimensional space for some constant d, but our results apply in the following more general setting. A metric space is said to have *doubling dimension* d if for any $r > 0$, a ball

of radius r can be covered by at most d balls of radius $r/2$ (see, e.g., [27]). A *doubling space* is a metric space of constant doubling dimension. Our results apply generally to metric spaces of constant doubling dimension, and it is well known that any Euclidean space of constant dimension is doubling. We model the objects of our system as points moving continuously in this space, and we make no assumptions *a priori* about the nature of this motion. Each sensor observes some *region* surrounding it. In general, our framework makes no assumptions about the size, shape, or density of these regions (though additional assumptions are imposed in Section 4.1 for analysis purposes). The sensor regions need not be disjoint, nor do they need to cover all the moving points at any given time.

Each sensor continually collects statistical information about the points lying within its region, and it outputs this information at synchronized time steps. As mentioned above, we assume throughout that this information is simply an *occupancy count* of the number of points that lie within the region. (The assumption of synchronization is mostly for the sake of convenience of notation. As we shall see, our compression algorithm operates jointly on local groups of a fixed size, and hence it is required only that the sensors of each group behave synchronously.)

As mentioned in the introduction, the pure form of our framework is based on an information-theoretic approach. Let us begin with a few basic definitions (see, e.g., [10]). We assume that the sensor streams can be modeled by a stationary, ergodic random process. Since the streams are synchronized and the cardinality of the moving point set is finite, we can think of the S sensor streams as a collection of S strings, each of length T, over a finite alphabet. The *entropy* of a discrete random variable X, denoted $\mathcal{H}(X)$, is defined to be $-\sum_x p(x)\log p(x)$, where the sum is over the possible values x of X, and $p(x)$ is the probability of x. (Throughout, logarithms are taken base 2.)

It is common to generalize entropy to random processes as follows. Given a stationary, ergodic random process X, consider the limit of the entropy of arbitrarily long sequences of X, normalized by the sequence length. This leads to the notion of *normalized entropy*, which is defined to be

$$\mathcal{H}(X) = \lim_{T\to\infty} -\frac{1}{T}\sum_{x,|x|=T} p(x)\log p(x),$$

where the sum is over sequences x of length T, and $p(x)$ denotes the probability of observing this sequence. Normalized entropy considers not only the distribution of individual characters, but the tendencies for certain patterns of characters to repeat over time.

Entropy can also be generalized to collections of random variables. Given a sequence $\mathbf{X} = \langle X_1, X_2, \ldots, X_S\rangle$ of (possibly statistically correlated) random variables, the *joint entropy* is defined to be $\mathcal{H}(\mathbf{X}) = -\sum_{\mathbf{x}} p(\mathbf{x})\log p(\mathbf{x})$, where the sum is taken over all S-tuples $\mathbf{x} = \langle x_1, x_2, \ldots, x_S\rangle$ of possible values, and $p(\mathbf{x})$ is the probability of this joint outcome [10]. The generalization to *normalized joint entropy* is straightforward. Normalized joint entropy further strengthens normalized entropy by considering correlations and statistical dependencies between the various streams.

In this paper we are interested in the lossless compression of the joint sensor stream. Shannon's source coding theorem states that in the limit, as the length of a stream of independent, identically distributed (i.i.d.) random variables goes to infinity, the minimum number of required bits to allow lossless compression of each character of the stream is equal to the entropy of the stream [38]. In our case, Shannon's theorem implies that the optimum bit rate of a lossless encoding of the joint sensor system cannot be less than the normalized joint entropy of the system. Thus, the normalized joint entropy is the "gold standard" for the asymptotic efficiency of any compression method. Henceforth, all references to "joint entropy" and "entropy" should be understood to mean the normalized versions of each.

In theory, optimal compression could be achieved in the limit by forming the joint system $\mathbf{X}$ described above and applying any entropy-based compression algorithm to the result. This approach would be hopelessly impractical, however, if the number of sensors is large. The reason is that the encoding optimality holds only in the limit, and the convergence rate degrades rapidly as the alphabet size increases. (This is because the number of possible strings of length w over an alphabet Σ grows as $|\Sigma|^w$. An entropy-based encoder, such as Lempel–Ziv sliding-window algorithm [44], must witness enough common patterns over its windows to effectively encode each of them.) The stream resulting by joining k streams, each with alphabet Σ_i, has a total alphabet of size $\prod_{i=1}^{k}|\Sigma_i|$, which would be unworkable if k is larger than a small constant. Instead, our approach will be to assume a limit on statistical dependencies among the observed sensor streams based on geometric locality. In particular, we will assume that each sensor's observation stream is statistically dependent on only a small number of neighboring sensors, which will make it possible to partition the sensors into local clusters, each of which can be compressed separately. (Later in Section 4.3 we will modify this assumption to allow a limited amount of dependence at larger distances.) Simply stated, this is the compression algorithm that we will introduce more formally in Section 5.2; partition the sensor streams into local neighborhoods and compress each neighborhood jointly, returning the set of these compressed neighborhood streams.

There are a number of natural ways to define neighboring sensors. One is an *absolute approach*, which is given a threshold distance parameter r, and in which it is assumed that any two sensors that lie at distance greater than r from each other have statistically independent observation streams. The second is a *relative approach* in which an integer m is provided, and it is assumed that two sensor observation streams are statistically dependent only if each is among the m nearest sensors of the other. In this paper we will take the latter approach. One reason is that it adapts to the local density of sensors. Another reason arises by observing that, in the absolute model, all the sensors might lie within distance r of each other. This means that all the sensors could be mutually statistically dependent, which would render optimal compression intractable. On the

other hand, if we deal with this by imposing the density restriction that no sensor has more than some number, say m, sensors within distance r, then the absolute approach reduces to a special case of the relative approach. (More in-depth theoretical and experimental analyses of the extent to which this assumption is reasonable can be found in Sections 4.1 and 4.2, respectively.)

This restriction allows reasoning about sensor streams in subsets. Previous restrictions of this form include the Lovász Local Lemma [14] which also assumes dependence on at most m events. Particle simulations (often used to simulate physical objects for animation) based on smoothed particle hydrodynamics have also used similar locality restrictions to determine which neighboring particles impact each other. These calculations are made over densely sampled particles and are based on a kernel function that determines the impact of one particle on another. This frequently amounts to a cut-off distance after which we assume that the particles are too far away to impact each other [1]. For a survey on smoothed particle hydrodynamics see [30].

Formally, let $P = \{p_1, p_2, \ldots, p_S\}$ denote the sensor positions. Given some integer parameter m, we assume that each sensor's stream can be statistically dependent on only its m nearest sensors. Since statistical dependence is a symmetric relation, two sensors can exhibit dependence only if each is among the m nearest neighbors of the other. More precisely, let $NN_m(i)$ denote the set of m closest sensors to p_i (not including sensor i itself). We say that two sensors i and j are *mutually m-close* if $p_i \in NN_m(j)$ and $p_j \in NN_m(i)$. A system of sensors is said to be *m-local* if for any two sensors that are not mutually m-close, their observations are statistically independent. (Thus, 0-locality means that the sensor observations are mutually independent.) Let $\mathbf{X} = \langle X_1, X_2, \ldots, X_S \rangle$ be a system of random streams associated with S sensors, and let $\mathcal{H}(\mathbf{X})$ denote its joint entropy. Given two random processes X and Y, the *conditional entropy* of X given Y is defined to be

$$\mathcal{H}(X \mid Y) = -\sum_{x \in X, y \in Y} p(x, y) \log p(x \mid y),$$

where $p(x, y)$ denotes the joint probability of both x and y occurring and $p(x \mid y)$ denotes the conditional probability that x occurs given that y occurs. Note that $\mathcal{H}(X \mid Y) \leq \mathcal{H}(X)$, and if X and Y are statistically independent, then $\mathcal{H}(X \mid Y) = \mathcal{H}(X)$ and generally $\mathcal{H}(X|Y) = \mathcal{H}(X, Y) - \mathcal{H}(Y)$. By the chain rule for conditional entropy [10], we have

$$\mathcal{H}(\mathbf{X}) = \mathcal{H}(X_1) + \mathcal{H}(X_2 \mid X_1) + \ldots + \mathcal{H}(X_i \mid X_1, \ldots, X_{i-1}) + \ldots + \mathcal{H}(X_S \mid X_1, \ldots, X_{S-1}).$$

Letting

$$D_i(m) = \{X_j : 1 \leq j < i \text{ and sensors } i \text{ and } j \text{ are mutually } m\text{-close}\}$$

we define the *m-local entropy*, denoted $\mathcal{H}_m(\mathbf{X})$, to be $\sum_{i=1}^{S} \mathcal{H}(X_i \mid D_i(m))$. Note that $\mathcal{H}(\mathbf{X}) \leq \mathcal{H}_m(\mathbf{X})$ and equality holds when $m = S$. By definition of m-locality, $\mathcal{H}(X_i \mid X_1, X_2, \ldots, X_{i-1}) = \mathcal{H}(X_i \mid D_m(i))$. By applying the chain rule for joint entropy, we have the following easy consequence, which states that, under our locality assumption, m-local entropy is the same as the joint entropy of the entire system.

Lemma 2.1. *Given an m-local sensor system with set of observations $\mathbf{X}$, $\mathcal{H}(\mathbf{X}) = \mathcal{H}_m(\mathbf{X})$.*

As mentioned earlier, the assumption of statistical independence is rather strong, since two distant sensor streams may be dependent simply because they exhibit a dependence with a common external event, such as the weather or time of day. Presumably, such dependencies would be shared by all sensors, and certainly by the m nearest neighbors. The important aspect of independence is encapsulated in the above lemma, since it indicates that, from the perspective of joint entropy, the m nearest neighbors explain essentially all the dependence with the rest of the system. In Section 4.3 we extend our understanding of independence to allow a limited dependence even among sensors that are not nearby.

One advantage of our characterization of mutually dependent sensor streams is that it naturally adapts to the distribution of sensors. It is not dependent on messy metric quantities, such as the absolute distances between sensors or the degree of overlap between sensed regions. Note, however, that our model can be applied in contexts where absolute distances are meaningful. For example, consider a setting in which each sensor monitors a region of radius r. Given two positive parameters α and β, we assume that the number of sensors whose centers lie within any ball of radius r is at most α, and the streams of any two sensors can be statistically dependent only if they are within distance βr of each other. Then, by a simple packing argument, it follows that such a system is m-local for $m = O(\alpha \beta^{O(1)})$, in any space of constant doubling dimension.

3. Entropy and independence

In order to understand and precisely examine the properties of the sensor observation streams, we must first consider some properties of entropy and independence in both the traditional statistical setting and the empirical setting. In this section, we determine some properties of the entropy of a stream consisting of the componentwise sum of two other streams. These lemmas will be useful when developing the compression algorithm in Section 5.

3.1. Statistical setting

We begin by reviewing basic definitions and results involving the entropy of a set of random processes in the traditional statistical setting (see, e.g., [10] for further information). Recall that in this setting a sensor's observation stream is modeled by a stationary, ergodic random process X over an alphabet Σ of fixed size. The *statistical probability* $p(x)$ of some outcome $x \in \Sigma$ is the probability associated with that outcome by the underlying random process. The *statistical entropy* of X is defined to be $-\sum_{x\in\Sigma} p(x)\log p(x)$. The *normalized statistical entropy* generalizes this to strings of increasing length:

$$\mathcal{H}_k(X) = -\frac{1}{k}\sum_{x\in\Sigma^k} p(x)\log p(x),$$

where in the standard definition, k is considered in the limit:

$$\mathcal{H}(X) = \lim_{k\to\infty} \mathcal{H}_k(X).$$

A fundamental fact from information theory is that this value represents a lower bound on the number of bits needed to encode a single character of the stream [10]. Unless otherwise specified, all references to *entropy* will mean normalized entropy. The *normalized joint statistical entropy* of two streams X and Y is defined to be

$$\mathcal{H}(X,Y) = \lim_{k\to\infty} -\frac{1}{k}\sum_{x,y\in\Sigma^k} p(x,y)\log p(x,y).$$

The normalized joint statistical entropy of a set of strings $\mathbf{X} = \{X_1, X_2, \ldots, X_Z\}$ is defined analogously and is denoted $\mathcal{H}(\mathbf{X})$.

We say that two sensor streams X and Y are *statistically independent* if, for all k and any $x, y \in \Sigma^k$, we have $p(x,y) = p(x)p(y)$. If X and Y are statistically independent then $\mathcal{H}(X,Y) = \mathcal{H}(X)+\mathcal{H}(Y)$ [10], and generally the joint entropy may be smaller. Also note that the componentwise sum of two streams carries less information than the two streams. The following technical lemma formalizes these two observations. The proof is straightforward, but for the sake of completeness we have included it in Appendix A.

Lemma 3.1. *Consider two sensor streams X and Y over the same time period. Let $X + Y$ denote the componentwise sum of these streams. Then $\mathcal{H}(X+Y) \le \mathcal{H}(X,Y) \le \mathcal{H}(X) + \mathcal{H}(Y)$.*

3.2. Empirical setting

Unlike statistical entropy, *empirical entropy* is based purely on the observed string, and does not assume an underlying random process. It replaces the probabilities of normalized entropy over substrings of length k by observed probabilities, conditioned on the value of the previous k characters. Let X be a string of length T over some alphabet Σ of fixed size. For $k \ge 1$ and $x \in \Sigma^k$ denoting a string of length k drawn from Σ, let $c_0(x)$ denote the number of times x appears in X, and let $c(x)$ denote the number of times x appears without being the suffix of X. Let $\mathsf{p}_X(x) = c(x)/(T-k)$ denote the *observed probability* of x in X. (When X is clear from context, we will express this as $\mathsf{p}(x)$.) Following the definitions of Kosaraju and Manzini [26], the *0th order empirical entropy* of a string X is defined to be

$$\mathsf{H}_0(X) = -\sum_{a\in\Sigma} \mathsf{p}(a)\log\mathsf{p}(a) = -\sum_{a\in\Sigma} \frac{c_0(a)}{T}\log\frac{c_0(a)}{T}.$$

For $a \in \Sigma$, let $\mathsf{p}_X(xa|x) = c(xa)/c(x)$ denote the observed probability that a is the next character of X immediately following x. The *kth order empirical entropy* is defined to be

$$\mathsf{H}_k(X) = -\frac{1}{T}\sum_{x\in\Sigma^k} c(x)\left[\sum_{a\in\Sigma} \mathsf{p}(xa|x)\log\mathsf{p}(xa|x)\right].$$

As observed in Kosaraju and Manzini [26], it is easily verified that $T\cdot\mathsf{H}_k(X)$ is a lower bound to the output size of any compressor that encodes each symbol with a code that only depends on the symbol itself and the k immediately preceding symbols. In the rest of this section, we introduce new extensions of these notions of empirical entropy to concepts that are analogous to those defined for the statistical entropy. Given two strings $X, Y \in \Sigma^T$ and $x, y \in \Sigma^k$, define $c(x,y)$ to be the count of the number of indices i, $1 \le i \le T-k$, such that $X[i\ldots i+k-1] = x$ and $Y[i\ldots i+k-1] = y$. Define $\mathsf{p}_{X,Y}(x,y) = c(x,y)/(T-k)$. For $a, b \in \Sigma$, define $\mathsf{p}_{X,Y}(xa, yb|x,y) = c(xa,yb)/c(x,y)$ to be the observed probability of seeing a and b in X and Y, respectively, just after seeing x and y. The *joint empirical entropy* of X and Y is defined to be

$$\mathsf{H}_k(X,Y) = -\frac{1}{T}\sum_{x,y\in\Sigma^k} c(x,y)\left[\sum_{xa,yb\in\Sigma} \mathsf{p}_{X,Y}(xa,yb|x,y)\log\mathsf{p}_{X,Y}(xa,yb|x,y)\right].$$

The joint empirical entropy of a set of strings $\mathbf{X} = \{X_1, \ldots, X_Z\}$ is defined analogously and is denoted $\mathsf{H}_k(\mathbf{X})$.

We define the *conditional empirical entropy* of two strings $X, Y \in \Sigma^T$ to be

$$\mathsf{H}_k(X|Y) = -\frac{1}{T} \sum_{x,y \in \Sigma^k} c(x,y) \sum_{a,b \in \Sigma} \mathsf{p}_{X,Y}(xa, yb|x, y) \log \mathsf{p}_{X,Y}(xa|yb),$$

where we define $\mathsf{p}_{X,Y}(xa|yb) = \mathsf{p}_{X,Y}(xa, yb|x, y)/\mathsf{p}_Y(yb|y)$ to be the probability that a directly follows x in X given that b directly follows y in Y.

We say that two strings X and Y are *empirically independent* if, for all $j \leq k+1$ and all $x, y \in \Sigma^j$, the observed probability of x occurring at the same time instant as y is equal to the product of the observed probabilities of each outcome individually, that is, $\mathsf{p}_{X,Y}(x, y) = \mathsf{p}_X(x)\mathsf{p}_Y(y)$. If X and Y are empirically independent then this also implies that, for $a \in \Sigma$ and $b \in \Sigma$, $\mathsf{p}_{X,Y}(xa, yb|x, y) = \mathsf{p}_X(xa|x)\mathsf{p}_Y(yb|y)$.

The following technical lemma provides a few straightforward generalizations regarding properties of statistical entropy to empirical entropy. The proof has been included for completeness in Appendix A.

Lemma 3.2. *Consider two strings $X, Y \in \Sigma^T$. Let $X + Y$ denote the componentwise sum of these strings.*

(i) *If X and Y are empirically independent, $\mathsf{H}_k(X, Y) = \mathsf{H}_k(X) + \mathsf{H}_k(Y)$.*
(ii) $\mathsf{H}_k(X, Y) = \mathsf{H}_k(X) + \mathsf{H}_k(Y|X)$.
(iii) $\mathsf{H}_k(X, Y) \leq \mathsf{H}_k(X) + \mathsf{H}_k(Y)$.
(iv) $\mathsf{H}_k(X + Y) \leq \mathsf{H}_k(X) + \mathsf{H}_k(Y)$.

4. Locality and limited independence

In this section we present two results in support of our framework's assumptions about m-locality and independence of non-local sensor streams, one theoretical and one empirical. We begin in Section 4.1 by examining the locality component of this assumption through a theoretical comparison of the efficiency of our framework to KDS. Then we consider the practicality of the locality assumption through experimental analysis in Section 4.2. Finally, we expand our initially strict independence assumption to a notion of limited independence (in both the statistical and empirical settings) in Section 4.3. This relaxation of the independence restriction also necessitates a return to, and expansion of, the entropy property lemmas given in Section 3.

4.1. Efficiency with respect to short-haul KDS

We believe that m-local entropy is a reasonable measure of the complexity of representing geometric motion. It might seem at first that any system that is based on monitoring the motion of a large number of moving objects by the incremental counts of a large number of sensors would produce such a huge volume of data that it would be utterly impractical as a basis for computation. Indeed, this is why compression is such an important ingredient in our framework. But, is it reasonable to assume that lossless compression can achieve the desired degree of data reduction needed to make this scheme competitive with purely prescriptive methods such as KDS? In this section, we consider a simple comparison, which suggests that lossless compression can achieve nearly the same bit rates as KDS would need to describe the motion of moving objects.

This may seem like comparing "apples and oranges," since KDS assumes precise knowledge of the future motion of objects through the use of flight plans. In contrast, our framework has no precise knowledge of individual point motions (only the occupancy counts of sensor regions) and must have the flexibility to cope with whatever motion is presented to it. Our analysis will exploit the fact that, if the motion of each point can be prescribed, then the resulting system must have relatively low entropy. To make the comparison fair, we will need to impose some constraints on the nature of the point motion and the sensor layout. First, to model limited statistical dependence we assume that points change their motion plans after traveling some local distance threshold ℓ. Second, we assume that sensor regions are modeled as disks of constant radius, and (again to limit statistical dependence) not too many disks overlap the same region of space. These assumptions are not part of our framework. They are just useful for this comparison.

Here we will assume that flight plans are linear and that motion is in the plane, but generalizations are not difficult. Let Q denote a collection of n moving objects over some long time period $0 \leq t \leq T$. We assume that the location of the ith object is broken into some number of linear *segments*, each represented by a sequence of tuples $(\mathbf{u}_{i,j}, \mathbf{v}_{i,j}, t_{i,j}) \in (\mathbb{Z}^2, \mathbb{Z}^2, \mathbb{Z}^+)$, which indicates that in the time interval $t \in (t_{i,j-1}, t_{i,j}]$, the ith object is located at the point $\mathbf{u}_{i,j} + t \cdot \mathbf{v}_{i,j}$. (Let $t_{i,0} = 0$.) We assume that all these quantities are integers and that the coordinates of $\mathbf{u}_{i,j}$, $\mathbf{v}_{i,j}$ are each representable with at most b bits. Let $\Delta_{i,j} = t_{i,j} - t_{i,j-1}$ denote the length of the jth time interval for the ith point.

In most real motion systems objects change velocities periodically. To model this, we assume we are given a locality parameter ℓ for the system, and we assume that the maximum length of any segment (that is, $\max_{i,j} \Delta_{i,j} \cdot \|\mathbf{v}_{i,j}\|$) is at most ℓ. Let s be the minimum number of segments that need to be encoded for any single object. Assuming a fix-length

encoding of the numeric values, each segment requires at least $4b$ bits to encode, which implies that the number of bits needed to encode the entire system of n objects for a single time step is at least

$$B_{\mathrm{KDS}}(n,\ell) \geq \frac{n \cdot s \cdot (4b)}{T}.$$

We call this the *short-haul KDS bit rate* for this system.

In order to model such a scenario within our framework, let P denote a collection of S sensors in the plane. Let us assume that each sensor region is a disk of radius λ. We may assume that the flight plans have been drawn according to some stable random process, so that the sensor observation streams satisfy the assumptions of stationarity and ergodicity. We will need to add the reasonable assumption that the sensors are not too densely clustered (since our notion of locality is based on m-nearest neighbors and not on an arbitrary distance threshold.) More formally, we assume that, for some constant $\gamma \geq 1$, any disk of radius $r > 0$ intersects at most $\gamma \lceil r/\lambda \rceil^2$ sensor regions. Let $\mathbf{X} = (X_1, X_2, \ldots, X_S)$ denote the resulting collection of sensor observation streams, and let $\mathcal{H}_m(n,\ell) \stackrel{\text{def}}{=} \mathcal{H}_m(\mathbf{X})$ denote the normalized m-local entropy of the resulting system. Our main result shows that the m-local entropy is within a constant factor of the short-haul KDS bit rate, and thus is a reasonably efficient measure of motion complexity.

Theorem 4.1. *Consider a short-haul KDS and the sensor-based systems defined above. Then for all sufficiently large m*

$$\mathcal{H}_m(n,\ell) \leq \left(\frac{4\ell}{\lambda}\sqrt{\frac{\gamma}{m}} + 1\right) B_{\mathrm{KDS}}(n,\ell).$$

Before giving the proof, observe that this implies that if the locality parameter m grows proportionally to $(\ell/\lambda)^2$, then up to constant factors we can encode the observed continuous motion as efficiently as its raw representation. That is, m should be proportional to the square of the number of sensors needed to cover each segment of linear motion. Note that this is independent of the number of sensors and the number of moving objects. It is also important to note that this is independent of the sensor sampling rate. Doubling the sampling frequency will double the size of the raw data set, but it does not increase the information content, and hence does not increase the system entropy.

Corollary 4.1.1. *By selecting $m = \Omega((\ell/\lambda)^2)$, we have $\mathcal{H}_m(n,\ell) = O(B_{\mathrm{KDS}}(n,\ell))$.*

Proof of Theorem 4.1. Consider an arbitrary moving object j of the system, and let $X_{i,j}$ denote the 0–1 observation counts for sensor i considering just this one object. Let us denote the associated single-object sensor system for j as $\mathbf{X}_{(j)} = (X_{1,j}, X_{2,j}, \ldots, X_{S,j})$. Clearly, $\mathcal{H}_m(\mathbf{X}) \leq \sum_{j=1}^{n} \mathcal{H}_m(\mathbf{X}_{(j)})$, since the latter is an upper bound on the joint m-local entropy of the entire system, and as shown in Lemma 3.1 for the special case of two streams the sum of observations cannot have greater entropy than the joint system.

Let s_j be the number of segments representing the motion of object j. Each segment is of length at most ℓ. Consider the per-object KDS bit-rate for object j, denoted $B_{\mathrm{KDS}}(j)$. Note that KDS considers the motion of each object individually, so $B_{\mathrm{KDS}} = \sum_{j=1}^{n} B_{\mathrm{KDS}}(j)$. KDS requires $4b$ bits per segment, so $B_{\mathrm{KDS}}(j) \geq \frac{4 \cdot b \cdot s_j}{T}$. Let $\ell' = (\lambda/4)\sqrt{m/\gamma}$. Clearly, $\ell' > 0$. Subdivide each of the s_j segments into at most $\lfloor \ell/\ell' \rfloor$ subsegments of length ℓ' and at most one of length less than ℓ'. Then there is a total of at most $s_j(\ell/\ell' + 1)$ subsegments.

We claim that the joint entropy of the sensors whose regions intersect each subsegment is at most $4b$. To see this, observe that there are 2^{4b} possible linear paths upon which the object may be moving, and each choice completely determines the output of all these sensors (in this single-object system). The entropy is maximized when all paths have equal probability, which implies that the joint entropy is $\log_2(2^{4b}) = 4b$. Recall that at most $\gamma \lceil r/\lambda \rceil^2$ sensor regions intersect any disk of radius r. Clearly, each subsegment can be covered by a disk of radius ℓ'. Therefore, at most $\gamma \lceil \ell'/\lambda \rceil = \gamma \lceil (1/4)\sqrt{m/\gamma} \rceil^2$ sensor regions can intersect some subsegment. We assert that all sensors intersecting this subsegment are mutually m-close. To see this, consider some sensors with sensing region centers c_1 and c_2 that intersect such a subsegment. Observe that, by the triangle inequality, the distance between c_1 and c_2 is at most $2\lambda + \ell'$. Since $\ell' = (\lambda/4)\sqrt{m/\gamma}$, by choosing $m \geq 16\gamma$ it follows that $\lambda \leq \ell'$, so $2\lambda + \ell' \leq 3\ell'$. Thus, for each sensor with center c_1 whose region overlaps this subsegment, the centers of the other overlapping sensor regions lie within a disk of radius $3\ell'$ centered at c_1. In order to establish our assertion, it suffices to show that the number of sensor centers lying within such a disk is at most m. Again recall that at most $\gamma \lceil r/\lambda \rceil^2$ sensor regions intersect any disk of radius r, so at most $\gamma \lceil (3/4)\sqrt{m/\gamma} \rceil^2$ sensor regions intersect the disk of radius $3\ell'$. Under our assumption that $m \geq 16\gamma$, it is easy to verify that $\gamma \lceil (3/4)\sqrt{m/\gamma} \rceil^2 \leq m$, as desired. Since the overlapping sensors are all mutually m-close, their m-local entropy is equal to their joint entropy, and so the m-local entropy is also at most $4b$. Thus, to establish an upper bound on $\mathcal{H}_m(\mathbf{X}_{(j)})$, it suffices to multiply the total number of subsegments by $4b$ and normalize by dividing by T. So we have

$$\mathcal{H}_m(\mathbf{X}_{(j)}) \leq \left(\frac{\ell}{\ell'} + 1\right)\frac{4bs_j}{T} \leq \left(\frac{\ell}{\ell'} + 1\right) B_{\mathrm{KDS}}(j) = \left(\frac{4\ell}{\lambda}\sqrt{\frac{\gamma}{m}} + 1\right) B_{\mathrm{KDS}}(j).$$

Considering the normalized m-local entropy of the entire system, we have

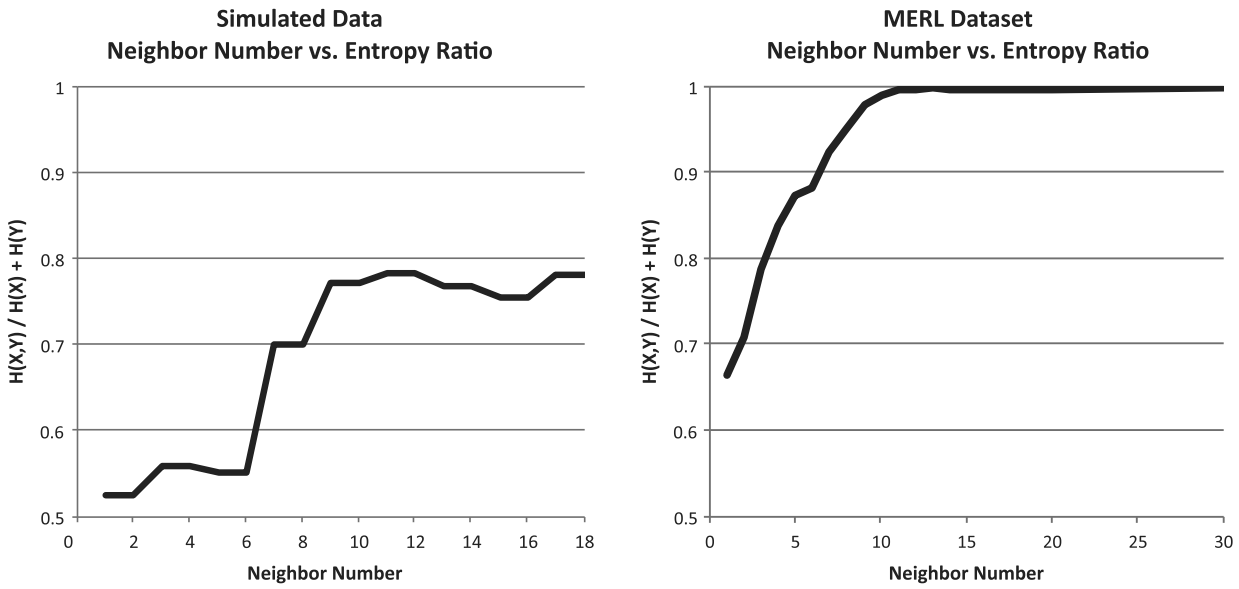

Fig. 2. A comparison of neighbor numbers and an entropy ratio indicating locality for 10 days of data. *Left*: Simulated data. *Right*: MERL hallway data.

$$\mathcal{H}_m(\mathbf{X}) \leq \sum_{j=1}^{n} \left(\frac{4\ell}{\lambda} \sqrt{\frac{\gamma}{m}} + 1 \right) B_{\mathrm{KDS}}(j) = \left(\frac{4\ell}{\lambda} \sqrt{\frac{\gamma}{m}} + 1 \right) B_{\mathrm{KDS}},$$

which completes the proof. □

4.2. Experimental locality results

In order to provide evidence that our framework's locality properties are satisfied on real-world data, we provide an empirical analysis of these assumptions. Although our data sets are rather small, they are large enough to demonstrate the value of local clustering in compression. We consider experimental results on two data sets – a simulated data set consisting of 19 sensors observing equal-sized portions of a circular highway (simulated using the "intelligent driver" traffic model [40, 41]) and a data set of hallway observations from the Mitsubishi Electronic Research Laboratories (MERL) consisting of 213 sensors [42]. Both of these data sets contain one count per second, representing the number of cars or people, respectively, within the sensor region during that second. For the MERL data set, these counts were derived from activation times (given by epoch time stamps) by considering each sensor activation to contribute a count of one to the associated sensor region for that second. We consider the assumption of m-locality in the context of the simulated data and the collected hallway movement data.

We examine the assumption that the sensor systems are m-local by considering the entropy relationship of a single sensor stream with that of its mth neighbor (where neighbor relationships are determined based on graph distance in the underlying network), for increasing values of m. Specifically, we consider the *entropy ratio*, $\mathsf{H}_k(X, Y)/(\mathsf{H}_k(X) + \mathsf{H}_k(Y))$, that is, the ratio of the pairwise joint empirical entropy (for $k = 4$) of the two sensors to the sum of their individual entropies for 10 days of data. This value can range from 0.5 to 1.0, and is low when two sensor streams are statistically dependent and increases to unity when the sensor streams are independent. As shown in Fig. 2, both the simulated data and the MERL data set have clear local neighborhoods where the entropy ratio is low, and as the neighbor number increases so does the entropy ratio. Our results show that, in both data sets, there is strong evidence in support of the underlying hypothesis of our framework, namely that nearby sensors have much lower joint entropy than distant ones.

4.3. Limited independence

Perfect statistical or empirical independence is too strong an assumption to impose on sensor observations. For example, if strings are drawn from independent sources, empirical independence will hold only in the limit. To deal with this, in this section we introduce a notion of limited independence for both the statistical and empirical settings. Given $0 \leq \delta < 1$, we say that a set of sensor streams $\mathbf{X} = \{X_1, X_2, ..., X_Z\}$ is *statistically* δ*-independent* if, for any k and outcomes $x_i \in \Sigma^k$,

$$(1 - \delta) \prod_{i=1}^{Z} p(x_i) \leq p(x_1, x_2, ..., x_Z) \leq (1 + \delta) \prod_{i=1}^{Z} p(x_i).$$

In the following lemma, we develop a relationship regarding the entropies of statistically δ-independent streams. This lemma is analogous to the property that $\mathcal{H}(\mathbf{X}) = \sum_{i=1}^{Z} \mathcal{H}(X_i)$ for mutually statistically independent streams. In Section 5.4 when we extend the compression algorithm to work in a δ-independent setting, we will need to make use of this property (and its empirical equivalent developed in Lemma 4.2) in order to determine a lower bound for a compression algorithm that correctly takes advantage of any dependence among the streams.

Lemma 4.1. *Given $0 \leq \delta < 1$ and a set of statistically δ-independent streams $\mathbf{X} = \{X_1, X_2, \ldots, X_Z\}$,*

$$(1-\delta)\left(\sum_{i=1}^{Z} \mathcal{H}(X_i)\right) - O(\delta) \leq \mathcal{H}(\mathbf{X}) \leq (1+\delta)\left(\sum_{i=1}^{Z} \mathcal{H}(X_i)\right) + O(\delta).$$

Proof. For simplicity of presentation, here we prove the lemma for the special case of just two sets $\mathbf{X} = \{X, Y\}$, but the generalization to multiple sets is straightforward.

Recall that

$$\mathcal{H}(X, Y) = \lim_{k\to\infty} -\frac{1}{k} \sum_{x\in X, y\in Y} p(x, y) \log p(x, y).$$

By the assumption of statistical δ-independence, and by manipulation of the definitions, we have

$$\begin{aligned}
\mathcal{H}(X, Y) &\leq \lim_{k\to\infty} -\frac{1}{k} \sum_{x\in X, y\in Y} p(x)p(y)(1+\delta) \log\big(p(x, y)\big) \\
&\leq \lim_{k\to\infty} \frac{1}{k} \sum_{x\in X, y\in Y} p(x)p(y)(1+\delta) \log \frac{1}{p(x)p(y)(1-\delta)} \\
&= \lim_{k\to\infty} \frac{1+\delta}{k}\left[-\sum_{x\in X}\big(p(x)\log p(x)\big) - \sum_{y\in Y}\big(p(y)\log p(y)\big) + \log\frac{1}{1-\delta}\right] \\
&= (1+\delta)\big(\mathcal{H}(X) + \mathcal{H}(Y)\big) + \lim_{k\to\infty} \frac{1+\delta}{k} \log\frac{1}{1-\delta}.
\end{aligned}$$

By a Taylor expansion in the neighborhood of $\delta = 0$, we see that $(1+\delta)\log\frac{1}{1-\delta} = O(\delta)$, which yields $\mathcal{H}(X, Y) \leq (1+\delta)(\mathcal{H}(X) + \mathcal{H}(Y)) + O(\delta)$. The proof that $(1-\delta)(\mathcal{H}(X) + \mathcal{H}(Y)) - O(\delta)$ follows symmetrically. □

We also introduce the idea of limited independence in the context of empirical entropy. Given $0 \leq \delta < 1$, a set of strings $\{X_1, X_2, \ldots, X_Z\}$ is *empirically δ-independent* if, for all $x_i \in \Sigma^j$ for $j \leq k+1$,

$$(1-\delta)\prod_{i=1}^{Z} \mathsf{p}(x_i) \leq \mathsf{p}(x_1, x_2, \ldots, x_Z) \leq (1+\delta)\prod_{i=1}^{Z} \mathsf{p}(x_i).$$

Lemma 4.2. *Given $0 \leq \delta < 1$, and a set of empirically δ-independent strings $\mathbf{X} = \{X_1, X_2, \ldots, X_Z\}$ for $X_i \in \Sigma^j$ where $j \leq k+1$,*

$$(1-\delta)\sum_{i=1}^{Z} \mathsf{H}_k(X_i) - O(\delta) \leq \mathsf{H}_k(\mathbf{X}) \leq (1+\delta)\sum_{i=1}^{Z} \mathsf{H}_k(X_i) + O(\delta).$$

Proof. As before, for simplicity of presentation, here we prove the lemma for the special case of just two sets $\mathbf{X} = \{X, Y\}$, but the generalization to multiple sets is straightforward.

$$\mathsf{H}_k(X, Y) = -\frac{1}{T} \sum_{x, y\in\Sigma^k} c(x, y) \sum_{xa, yb\in\Sigma} \mathsf{p}(xa, yb|x, y) \log \mathsf{p}(xa, yb|x, y),$$

where $\mathsf{p}(xa, yb|x, y) = \mathsf{p}_{X,Y}(xa, yb|x, y)$. Recall that $\mathsf{p}_{X,Y}(xa, yb|x, y) = c(xa, yb)/c(x, y)$ where $c(xa, yb)$ is the number of times the string $xa \in X$ appears at the same indices as $yb \in Y$. We have

$$\begin{aligned}
\mathsf{H}_k(X, Y) &= -\frac{1}{T} \sum_{x, y\in\Sigma^k} c(x, y) \sum_{a,b\in\Sigma} \frac{c(xa, yb)}{c(x, y)} \log \mathsf{p}(xa, yb|x, y) \\
&= -\frac{1}{T} \sum_{x, y\in\Sigma^k} \sum_{a,b\in\Sigma} \frac{c(xa, yb)(T-k)}{T-k} \log \mathsf{p}(xa, yb|x, y).
\end{aligned}$$

Since $\mathsf{p}(xa, yb) = \frac{c(xa,yb)}{T-k}$, $\mathsf{p}(x, y) = \frac{c(x,y)}{T-k}$, and $\mathsf{p}(xa, yb|x, y) = \frac{c(xa,yb)}{c(x,y)}$ this is

$$\begin{aligned}\mathsf{H}_k(X, Y) &= -\frac{T-k}{T} \sum_{x,y\in\Sigma^k} \sum_{a,b\in\Sigma} \mathsf{p}(xa, yb) \log\left(\frac{(T-k)\cdot c(xa, yb)}{(T-k)\cdot c(x, y)}\right) \\ &= -\frac{T-k}{T} \sum_{x,y\in\Sigma^k} \sum_{a,b\in\Sigma} \mathsf{p}(xa, yb) \log\left(\frac{\mathsf{p}(xa, yb)}{\mathsf{p}(x, y)}\right).\end{aligned}$$

Before proceeding with this analysis, we develop a useful relationship.

$$\begin{aligned}&-\frac{(T-k)}{T}\left[\sum_{x,y\in\Sigma^k} \mathsf{p}(x)\mathsf{p}(y) \sum_{a,b\in\Sigma} \mathsf{p}(xa|x)\mathsf{p}(yb|y) \log\big(\mathsf{p}(xa|x)\mathsf{p}(yb|y)\big)\right] \\ &= -\frac{1}{T}\left[\sum_{x\in\Sigma^k} c(x) \sum_{a\in\Sigma} p(xa|x) \log p(xa|x) + \sum_{y\in\Sigma^k} c(y) \sum_{b\in\Sigma} p(yb|y) \log p(yb|y)\right] \\ &= \mathsf{H}_k(X) + \mathsf{H}_k(Y).\end{aligned}$$

Now we develop an upper bound on the earlier equation. Let $f = -\mathsf{p}(xa, yb) \log \frac{\mathsf{p}(xa,yb)}{\mathsf{p}(x,y)} = \mathsf{p}(xa, yb) \log \frac{\mathsf{p}(x,y)}{\mathsf{p}(xa,yb)}$. Then the equation we wish to bound is

$$\frac{T-k}{T} \sum_{x,y\in\Sigma^k} \sum_{a,b\in\Sigma} f,$$

where, by the definition of δ-independence,

$$f \le (1+\delta)\mathsf{p}(xa)\mathsf{p}(yb) \log\left(\frac{\mathsf{p}(x, y)}{\mathsf{p}(xa, yb)}\right) \le (1+\delta)\mathsf{p}(xa)\mathsf{p}(yb) \log\left(\frac{(1+\delta)\mathsf{p}(x)\mathsf{p}(y)}{(1-\delta)\mathsf{p}(xa)\mathsf{p}(yb)}\right).$$

Since $\mathsf{p}(xa, yb) = \mathsf{p}(xa|x)\mathsf{p}(x)\mathsf{p}(yb|y)\mathsf{p}(y)$, this is equal to

$$\begin{aligned}&(1+\delta)\mathsf{p}(x)\mathsf{p}(y)\mathsf{p}(xa|x)\mathsf{p}(yb|y) \log\left(\frac{(1+\delta)}{(1-\delta)\mathsf{p}(xa|x)p(yb|y)}\right) \\ &= (1+\delta)\mathsf{p}(x)\mathsf{p}(y)\mathsf{p}(xa|x)\mathsf{p}(yb|y)\left(\log\frac{(1+\delta)}{(1-\delta)} - \log\big(\mathsf{p}(xa|x)\mathsf{p}(yb|y)\big)\right).\end{aligned}$$

Substituting back in for f and using our previously developed relationship, we have

$$\begin{aligned}\mathsf{H}_k(X, Y) &\le (1+\delta)\big(\mathsf{H}_k(X) + \mathsf{H}_k(Y)\big) \\ &\quad + \frac{(1+\delta)(T-k)}{T} \sum_{x,y\in\Sigma^k} \mathsf{p}(x)\mathsf{p}(y) \sum_{a,b\in\Sigma} \mathsf{p}(xa|x)\mathsf{p}(yb|y) \log\frac{1+\delta}{1-\delta} \\ &= (1+\delta)\big(\mathsf{H}_k(X) + \mathsf{H}_k(Y)\big) + \frac{(1+\delta)(T-k)}{T} \log\frac{1+\delta}{1-\delta}.\end{aligned}$$

Let

$$g(\delta) = \log\frac{1+\delta}{1-\delta} = \log\left(1 + \frac{2\delta}{1-\delta}\right).$$

Consider the Taylor expansion for $g(\delta)$ in the neighborhood of $\delta = 0$ (i.e., the Maclaurin series). The Maclaurin series for $g(\delta)$ is within a constant factor of the expansion for $\log(1/(1-\delta)) = \delta + \delta^2/2 + \delta^3/3 + O(\delta^4)$. Since $\delta < 1$ by definition, $\delta^i > \delta^j$ for $i < j$, so $\log(1/(1-\delta)) = O(\delta)$ and $g(\delta) = O(\delta)$. Substituting back into our main inequality, we have

$$\begin{aligned}\mathsf{H}_k(X, Y) &\le (1+\delta)\big(\mathsf{H}_k(X) + \mathsf{H}_k(Y)\big) + \frac{(1+\delta)(T-k)}{T} O(\delta) \\ &\le (1+\delta)\big(\mathsf{H}_k(X) + \mathsf{H}_k(Y)\big) + O(\delta).\end{aligned}$$

The proof that

$$(1-\delta)\big(\mathsf{H}_k(X) + \mathsf{H}_k(Y)\big) - O(\delta) \le \mathsf{H}_k(X, Y)$$

proceeds symmetrically. □

***Partition*(point set P, m)**

```
for all p ∈ P                          // Determine the m nearest neighbors and the radius
    determine NN_m(p) and r_m(p)       // of the m nearest neighbors ball based on the original
i = 1                                  // point set. (These values do not change.)
while P ≠ ∅                            // While unpartitioned points remain
    unmarked(P) = P                    //     Unmark all remaining points.
    P_i = ∅                            //     Create a new, empty partition.
    while unmarked(P) ≠ ∅              //     While unmarked points remain
        r = min_{p∈unmarked(P)} r_m(p) //         Find the point p with the minimum radius (r)
        p' = p ∈ P : r = r_m(p)        //         nearest neighbor ball and add that point and
        P_i = P_i ∪ {p ∈ P : ‖pp'‖ ≤ r} //        add all points within r to the new partition.
        P = P \ {p ∈ P : ‖pp'‖ ≤ r}    //         Remove these points from P and mark
        unmarked(P) = unmarked(P) \ {p ∈ unmarked(P) : ‖pp'‖ ≤ 3r}
    increment i                        //         points within distance 3r of p.
return {P_1, P_2, ..., P_c}            // Return the resulting partitions.
```

Fig. 3. The partitioning algorithm employed in the proof of Lemma 5.1.

5. Compression results

Having developed the necessary framework and associated realistic modifications and analyses, we can now introduce the main compression results within this framework. The principal result of this section is that given the observation streams of a system of sensors in our framework, it is possible to represent the combined streams in a manner that is within a constant factor of the information-theoretic lower bound. As mentioned in Section 2, we will exploit the fact that the sensor system is m-local in order to partition the sensors into a small number of sets such that each set can be further partitioned into clusters that are pairwise statistically independent. Since the clusters are independent, they can each be compressed individually.

In Section 5.1 we establish the main partitioning result upon which our approach is based. In Section 5.2 we present the compression algorithm within the pure version of our framework. Later in Section 5.4 we generalize this to the more realistic empirical context and under the assumption of limited independence.

5.1. Partitioning algorithm

Before presenting our partitioning results, we introduce some definitions regarding properties of the static point set representing sensor locations. Let P denote an n-element point set in $\mathbb{R}^d$ representing the locations of a set of sensors. Let us make the general-position assumption that the distances between all pairs of points of P are distinct, so that mth nearest neighbors can be defined uniquely. Given an integer parameter m and $p \in P$, let $r_m(p)$ denote the distance from p to its mth nearest neighbor in $P \setminus \{p\}$. Recall from Section 2 that two points $p, q \in P$ are mutually m-close if they are each among the other's m nearest neighbors, that is $\|pq\| \leq \min(r_m(p), r_m(q))$. Throughout, we will assume that m is a constant (which may depend on the dimension, but not on n). We say that a subset $P' \subseteq P$ is *m-clusterable* if it can be partitioned into subsets $C_{i1}, C_{i2}, \ldots$, called *clusters*, such that $|C_{ij}| \leq m+1$ and if p and q are mutually m-close then p and q are in the same subset of the partition. (Note that the definition of $r_m(p)$ used in this definition is with respect to P, not P'.) Intuitively, this means that naturally defined clusters in the set are sufficiently well separated so that points within the same cluster are closer to each other than they are to points outside of the cluster. The following lemma shows that in any space of constant doubling dimension (recall the definition from Section 2) it is possible to partition the set into subsets that are each *m-clusterable*.

Lemma 5.1. *In any doubling space there exists an integral constant c (depending on dimension) such that for any integral $m > 0$ and any set P in this space, P can be partitioned into at most c subsets, $P_1, P_2, \ldots$ each of which is m-clusterable.*

Proof. The partitioning can be computed using the algorithm *Partition*(P, m) given in Fig. 3. It works in a series of phases. At the start of each phase, all the points of P are unmarked. We repeatedly select the unmarked point $p \in P$ that minimizes $r = r_m(p)$. All the points of P that lie within distance r of p are added to the current partition and are removed from P. These removed points constitute a *cluster*, and p is called a *cluster center*. The points of P that lie within distance $3r$ of p are marked. (Intuitively, these points form a buffer region around the cluster.) The phase ends when all points have been marked, and the algorithm terminates when P is empty. Thus, each iteration of the inner loop creates a cluster, and each iteration of the outer loop creates a subset of the partition.

First, we show that the output P_i of each phase is an m-clusterable subset of P. Let $C_1, C_2, \ldots$ denote the clusters constituting P_i. Let $p_j \in P_i$ denote the cluster center for C_j. The points of the cluster are some subset of points of P lying within distance $r_j = r_m(p_j)$ of p_j, and therefore $|C_j| \leq m+1$. (Note that there may be fewer than $m+1$ points in the cluster,

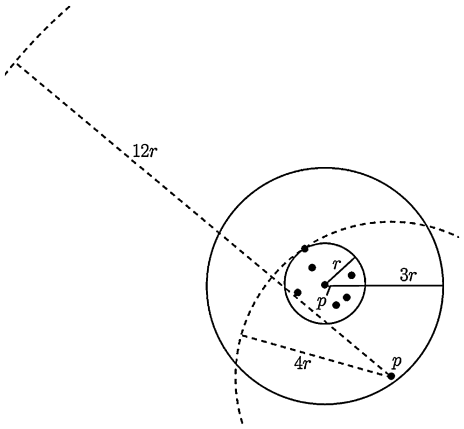

Fig. 4. Proof of Lemma 5.1 for $m = 6$.

because some of these points may have been removed from P earlier in the algorithm, and r_m is defined with respect to the original set P.) It suffices to show that for any $k > j$, no point $p \in C_j$ can be mutually m-close to a point $q \in C_k$. By construction, p lies within distance r_j of p_j, and therefore by the triangle inequality, p lies within distance $2r_j$ of at least m points of (the original set) P, namely p_j and $m-1$ of p_j's nearest neighbors (not counting p itself). This implies that $r_m(p) \le 2r_j$. By our marking process and the fact that $k > j$, any point $q \in C_k$ lies at distance greater than $3r_j$ from p_j, and so by the triangle inequality, q lies at distance greater than $3r_j - r_j = 2r_j$ from p. This implies that

$$\|pq\| > 2r_j \ge r_m(p) \ge \min\bigl(r_m(p), r_m(q)\bigr).$$

That is, p and q are not mutually m-close, as desired.

Next, we bound the number of clusters c in each partition. We refer to each iteration of the outer while loop as a *phase*. Note that at the end of the first phase, all points are either marked or removed from P. Consider some point p that remains after the first phase, and let p' be the point that marked it. Let $r = r_m(p')$ (see Fig. 4). By our marking process, p lies within distance $3r$ of p'. Since there are m points of P within distance r of p', by the triangle inequality, there are at least m points within distance $4r$ of p, so $r_m(p) \le 4r$. Let i denote the phase in which p is finally added to some cluster. It must have been marked by $i-1$ points at each of the earlier phases. Let $M(p)$ denote these points. In order to bound the number of sets in the partition, we will bound the number of times each point is marked by establishing an upper bound on $|M(p)|$.

Since clusters are formed in increasing order of r_m values, in order for a point q to mark p, it must have been chosen as a cluster center before p, implying that $r_m(q) < r_m(p)$. Therefore, any point q of $M(p)$ lies within distance

$$3 \cdot r_m(q) \le 3 \cdot r_m(p) \le 3 \cdot 4r = 12r$$

of p. Since p' was the first point to mark p, for all $q \in M(p)$ we have $r_m(q) \ge r_m(p) = r$. Whenever a point q is chosen as a cluster center, all the remaining points of P lying within distance $r_m(q) \ge r$ are removed from P and can never be a cluster center. Therefore, for each $q, q' \in M(p)$, we have $\|qq'\| > r$. In summary, the points of $M(p)$ all lie within a ball of radius $12r$ and each pair of points is separated by a distance of at least r. By a straightforward packing argument, in any doubling space the number such points is bounded by a function of the doubling dimension. Therefore, there exists a constant c (depending on the doubling dimension) such that $|M(p)| \le c-1$. Since no point can be marked more than $c-1$ times, every points will be placed in some cluster by phase c. □

5.2. Compression theorem

We now present the main compression algorithm and its analysis. Let us begin with some notation. Let $P = \{p_1, p_2, \ldots, p_S\}$ be an m-local system of sensors, and let $\mathbf{X} = \{X_1, X_2, \ldots, X_S\}$ be the observation streams recorded by these sensors. Let $\{P_1, P_2, \ldots, P_c\}$ denote the partition of P resulting by applying Lemma 5.1, and for $1 \le i \le c$, let $\mathbf{X}_i$ denote the collection of observation streams associated with P_i. Let $\{C_{ij}\}$ be the set of clusters associated with P_i, and let $\{\mathbf{X}_{ij}\}$ denote the corresponding clusters of observation streams. Letting h_{ij} denote the cardinality of C_{ij}, for $1 \le h \le h_{ij}$, let X_{ijh} be the hth stream in cluster C_{ij}. Finally, for $1 \le t \le T$, let X_{ijht} denote the tth observation of this stream.

Our compression algorithm first applies the partitioning algorithm of Lemma 5.1 (recall Fig. 3). It then jointly compresses the sensor streams of each cluster separately and returns their union. In particular, for each cluster C_{ij}, we create a string $\widehat{X}_{ij}$ whose tth character is a tuple consisting of the tth characters of each of the streams of C_{ij}. That is, $\widehat{X}_{ij} = \langle (X_{ij1t}, X_{ij2t}, \ldots, X_{ijh_{ij}t}) \rangle_{t=1}^{T}$. By Lemma 5.1, the points of P_i that lie in different clusters are not mutually m-close, and therefore their associated strings are statistically independent. It follows from independence that the joint entropy of $\mathbf{X}_i$ is

***PartitionCompress*(stream set X, sensor set P, m)**

$\{P_1, P_2, \ldots, P_c\} = Partition(P, m)$
for $i = 1$ to c
  for each cluster C_{ij} in P_i
    Let $\{X_{ij1}, \ldots, X_{ijh_{ij}}\}$ denote the streams corresponding to this cluster
    $\widehat{X}_{ij} = \langle (X_{ij1t}, X_{ij2t}, \ldots, X_{ijh_{ij}t}) \rangle_{t=1}^{T}$
return $\bigcup_{ij} entropy\_compress(\widehat{X}_{ij})$

Fig. 5. The *PartitionCompress* algorithm, which takes a set X of streams of length T and the associated set P of sensors which recorded them and returns a compressed encoding of the streams. The partitioning algorithm is given in Fig. 3 and computes the partitioning of P into subsets P_i and clusters C_{ij}. The function *entropy_compress* is any entropy-based compression algorithm.

the sum of the joint entropies of the string sets associated with the individual clusters, that is, $\mathcal{H}(\mathbf{X}_i) = \sum_{j=1}^{h_{ij}} \mathcal{H}(\mathbf{X}_{ij})$. By definition, the joint entropy of $\mathbf{X}_{ij}$ is equal to the entropy of the string $\widehat{X}_{ij}$. Therefore, we have

$$\mathcal{H}(\mathbf{X}_i) = \sum_{j=1}^{h_{ij}} \mathcal{H}(\widehat{X}_{ij}).$$

By applying an optimal entropy-based compression algorithm (for example, the Lempel–Ziv sliding-window compression algorithm [44]) to each of the cluster strings $\widehat{X}_{ij}$ for $1 \le j \le h_{ij}$, the union of these compressed strings is a faithful representation of the system $\mathbf{X}_i$ that is of optimal length (in the limit). Finally, the stream system $\mathbf{X}_i$ is a subset of the complete system $\mathbf{X}$, we have $\mathcal{H}(\mathbf{X}_i) \le \mathcal{H}(\mathbf{X})$. Thus, if we apply this to each sensor stream $\mathbf{X}_i$ associated with each subset P_i of the partition, we obtain $c = O(1)$ compressed streams, each of which can be represented using no more bits than an optimum encoding of the entire system.

The complete compression algorithm is presented in Fig. 5. Recall from Lemma 5.1 that each cluster is of size at most $m+1$, and therefore (by our assumption that m is a constant), the resulting strings $\widehat{X}_{ij}$ are over an alphabet that is a factor of at most $m+1$ times larger than the alphabets associated with the streams of the individual sensors. Therefore, in contrast to the approach of compressing the entire sensor system, this local approach is much more practical.

In order to state our main result we introduce some notation. Given a stream X and an entropy-based compression algorithm *alg*, let $\mathrm{Enc}_{alg}(X)$ denote the length (in bits) of the output of *alg* on input X. When the exact choice of compression algorithm is unimportant, we use $\mathrm{Enc}_{opt}(X)$ to denote the length of an ideal optimal entropy-based compression algorithm. Given a set of streams $\mathbf{X}$, let $\mathrm{Enc}_{alg}(\mathbf{X})$ denote the sum of the lengths resulting by applying the compression algorithm to each stream of $\mathbf{X}$ individually. Given a set of streams $\mathbf{X}$, let $S_{opt}(\mathbf{X})$ denote the optimal size (in bits) of an entropy-based encoding of $\mathbf{X}$. (We will rely on context to distinguish between $S_{opt}(\mathbf{X})$ in statistical and empirical contexts.) Our main compression result in the statistical context follows as an immediate consequence of the above.

Theorem 5.1. *In any doubling space there exists an integral constant c (depending on dimension) such that given any integral $m > 0$, an m-local sensor system P in this space, and an associated set of observation streams $\mathbf{X}$, it is possible to faithfully represent $\mathbf{X}$ as a collection of c streams $\{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_c\}$, such that:*

(i) *for* $1 \le i \le c$, $\mathrm{Enc}_{opt}(\mathbf{X}_i) \le S_{opt}(\mathbf{X})$,
(ii) $\mathrm{Enc}_{opt}(\mathbf{X}) \le c \cdot S_{opt}(\mathbf{X})$.

5.3. Experimental compression results

Next, we consider experimental results on the *PartitionCompress* algorithm. We compare the original observations size, the total size of all LZ78 dictionaries created for individual sensors' observations, and the size of the LZ78 dictionaries created according to the *PartitionCompress* algorithm taking advantage of dependence between neighboring sensors. We again consider the MERL data, this time for 10 hours of data, and consider neighbor numbers from the perspective of sensor "369," a sensor in the middle of a straight segment of hallway. The results, shown in Fig. 6, show a roughly 50-fold improvement of *PartitionCompress* over the uncompressed data and a 2-fold improvement over the data compressed without clustering, when $m = 5$. Note also that these results again confirm our locality assumptions, since the improvement increases until $m = 5$ at which point including more sensors in the locality neighborhood has no effect.

Recall from the previous sections that we proved that *PartitionCompress* creates $c = O(1)$ partitions ($c = 1 + 12^d$ for points in $\mathbb{R}^d$) and that *PartitionCompress* compresses the sensor system to within c times the optimal joint entropy bound. For the simulated data, when considered over all possible values of m, c ranged from 1 to 4 with a median value of 3. For the MERL data set, when considered over values of m from 1 to 50, c ranged from 3 to 6 with a median value of 5. In comparison, the worst case bound gives a value of $c = 145$ for two-dimensional Euclidean space. Thus, c is shown in

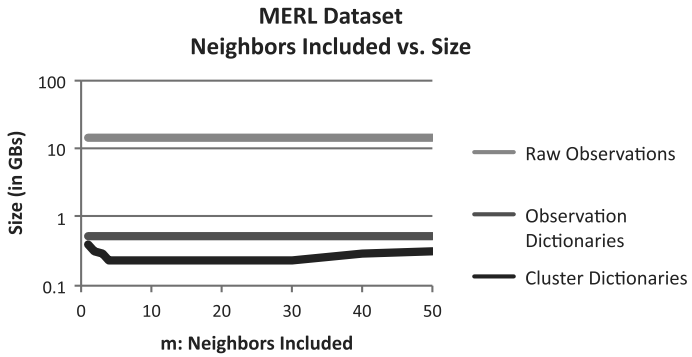

Fig. 6. A comparison of the size required for 10 hours of MERL data when considered raw, compressed using LZ78 dictionaries on individual sensors, or compressed using the per-cluster dictionaries created by *PartitionCompress*.

practice to be much less than the worst case bound, and the compressed size of the data achieved by *PartitionCompress* is correspondingly only a small factor away from the optimal.

5.4. Compression with limited independence

In this section we will consider the encoding size that can be achieved by *PartitionCompress* when analyzed in terms of empirical entropy and when the framework is extended to allow δ-independence. Since *PartitionCompress* relies on a compression algorithm as a subroutine, the compression bounds for this subroutine will impact the final encoding size achieved by *PartitionCompress*. We will analyze this size in both statistical and empirical settings. In either context, we will use $\text{Enc}_{alg}(\mathbf{X})$ to denote the length of the encoded set of sensor streams $\mathbf{X}$, where *alg* is the compression algorithm used by *PartitionCompress*.

5.4.1. Statistical setting

Given a set of statistically independent streams $\mathbf{X} = \{X_1, X_2, \ldots, X_S\}$ in a statistical setting, standard information theory results [10] tell us that the optimal encoded space is $\sum_{i=1}^{S} \mathcal{H}(X_i)$ bits. Recall that we denote this $S_{opt}(\mathbf{X})$. From Section 4.3, we know that the optimal space used by an encoded set of statistically δ-independent streams $\mathbf{X}$ is $(1-\delta)(\sum_{i=1}^{S} \mathcal{H}(X_i)) - O(\delta)$ bits. Call this $S_{opt}(\mathbf{X}, \delta)$. Let *opt* be some compression algorithm that achieves the optimal statistical entropy encoding length, for example the Lempel–Ziv trie-based encoding algorithm that we will refer to as LZ78 [45]. We know from Section 5.2 that $\text{Enc}_{opt}(\mathbf{X}) = O(\mathcal{H}(\mathbf{X}))$ bits for a set of observations from an m-local sensor system, where the hidden constant is exponential in the doubling dimension. We define a *statistically* (δ, m)*-local sensor system* to be the same as an m-local sensor system but with an assumption of δ-independence between the clusters instead of pure independence. We have the following theorem regarding the space used by *PartitionCompress*:

Theorem 5.2. *Given a set* $\mathbf{X}$ *of sensor streams from a statistically* (δ, m)*-local sensor system, for any* $0 \leq \delta < 1 - \varepsilon$*, where* δ *may vary based on the sensor system and* $\varepsilon > 0$ *is an absolute constant,*

$$\text{Enc}(\mathbf{X}) = O\left(\max\left\{\delta T, S_{opt}(\mathbf{X})\right\}\right) \textit{ bits}.$$

Proof. An optimal algorithm would compress each partition to take the greatest advantage of the dependence between clusters. It would achieve a space bound of

$$S_{opt}(\mathbf{X}, \delta) = T(1-\delta)\left[\sum_{i=1}^{S} \mathcal{H}(X_i)\right] - T \cdot O(\delta)$$

for each partition. The *PartitionCompress* algorithm compresses each partition to space

$$S_{opt}(\mathbf{X}) = T \sum_{i=1}^{S} \mathcal{H}(X_i).$$

The ratio is

$$\rho = \frac{S_{opt}(\mathbf{X})}{S_{opt}(\mathbf{X}, \delta)} = \frac{\sum_{i=1}^{S} \mathcal{H}(X_i)}{(1-\delta)[\sum_{i=1}^{S} \mathcal{H}(X_i)] - O(\delta)}.$$

Here we consider the two possible cases for the relationship of $O(\delta)$ to $\sum_{i=1}^{S} \mathcal{H}(X_i)$:

1. *Case* $O(\delta) \geq \sum_{i=1}^{S} \mathcal{H}(X_i)$*:*

$$\frac{\sum_{i=1}^{S} \mathcal{H}(X_i)}{(1-\delta)[\sum_{i=1}^{S} \mathcal{H}(X_i)] - O(\delta)} = \left[\frac{1}{1-\delta}\right] O(\delta) = O(\delta).$$

   For this case, *PartitionCompress*'s space bound is within $O(\delta)$ of the optimal for a single one of the c partitions, or a total of $O(\delta)$ times $S_{opt}(\mathbf{X}, \delta)$, which is $O(\delta \cdot T)$, since $O(\delta) \geq \sum_{i=1}^{S} \mathcal{H}(X_i)$.
2. *Case* $O(\delta) < \sum_{i=1}^{S} \mathcal{H}(X_i)$*:*

$$\frac{1}{1-\delta}\left[\frac{1}{1 - O(\delta)/(1-\delta)\sum_{i=1}^{S} \mathcal{H}(X_i)}\right] \leq \frac{1}{1-\delta}\left[\frac{1}{1-\frac{1}{1-\delta}}\right] = O\left(\frac{1}{1-\delta}\right) = O(1+\delta).$$

   For this case, *PartitionCompress*'s space bound is $O((1+\delta)S_{opt}(\mathbf{X}, \delta))$, which is $O(S_{opt}(\mathbf{X}))$ since $\delta < 1 - \varepsilon$.

The final total space bound is $\max\{O(\delta T), O(S_{opt}(\mathbf{X}))\}$. □

The space established in Theorem 5.2 is the basic statistical encoded space bound. It hides constants that are exponential in m and the doubling dimension.

5.4.2. Empirical setting

In the rest of this section, we extend the statistical results from the earlier part of this section to the empirical setting. In order to reason about the empirically optimal space bound for a set of strings $\mathbf{X}$, consider the string X^* over the alphabet Σ^S created from the original set of strings by letting the ith character of the new string, for $1 \leq i \leq T$, be equal to a new character created by concatenating the ith character of each string in the original set. As mentioned earlier, the new string's optimal encoded space bound is $T \cdot \mathsf{H}_k(X^*)$.

Lemma 5.2. *Given a set of strings* $\mathbf{X}$ *and a string* X^* *created from* $\mathbf{X}$ *as described above,* $\mathsf{H}_k(X^*) = \mathsf{H}_k(\mathbf{X})$.

Proof. Recall that the definition of joint empirical entropy is based on the observed probability that single characters occur in all strings at the same string index directly after specific substrings of length k. Observe that by the construction of X^*, simultaneous occurrences appear for the same indices at which a single combined character appears in X^*. This observation implies that if $\mathsf{H}_k(\mathbf{X})$ is restated to refer to the characters appearing in X^*, $\mathsf{H}_k(X^*) = \mathsf{H}_k(\mathbf{X})$. □

Corollary 5.2.1. *The minimum number of bits to encode a set* $\mathbf{X}$ *of strings, assuming that each character depends only on the preceding* k *characters, is* $S_{opt}(\mathbf{X}) = T \cdot \mathsf{H}_k(\mathbf{X})$.

Although this construction suggests a compression procedure, it is impractical because in order to capture the repetitive nature of the strings in $\mathbf{X}$, the window size k would need to grow exponentially based on the size of the alphabet for each additional sensor stream. Instead, we use the more local approach of *PartitionCompress*.

We define an *empirically m-local sensor system* to be analogous to the definition of an m-local sensor system, but with an assumption of empirical independence instead of statistical independence. Similarly, an *empirically* (δ, m)*-local sensor system* assumes empirical δ-independence instead of statistical independence. The algorithm *PartitionCompress* relies on an entropy encoding algorithm as a subroutine. In the context of an empirical entropy-based analysis it would be appropriate to use the data structure developed by Ferragina and Manzini [15] as the subroutine that jointly compresses the streams from a single cluster. The Ferragina and Manzini structure [15] gives an optimal space bound of $O(T \cdot \mathsf{H}_k(X_i)) + T \cdot o(1)$ where X_i is the merged stream for that single cluster. We are interested in developing a lower bound on the compression that can be achieved using *PartitionCompress* in an empirical setting. Instead of using a specific algorithm we use the bound of $S_{opt}(\mathbf{X})$ discussed earlier and call the algorithm that achieves this bound *opt*. Assuming empirical independence of the set of strings $\mathbf{X}$ from S separate clusters within a single partition, compressing these clusters separately achieves the optimal bound of $S_{opt}(\mathbf{X}) = T \cdot \sum_{i=1}^{S} \mathsf{H}_k(X_i)$ space for a single partition. As a direct consequence, we have the following theorem.

Theorem 5.3. *Let* $\mathbf{X}$ *denote a set of sensor streams from an empirically m-local sensor system. Then* $\mathrm{Enc}_{opt}(\mathbf{X})$ *consists of* $O(S_{opt}(\mathbf{X}))$ *bits.*

The hidden constants from Theorem 5.3 and for Theorems 5.4 and 5.5 grow exponentially in m and the doubling dimension of the space in which the sensors reside. If we consider empirical δ-independence, then the lower bound achieved by the compression algorithm (over S total clusters in all partitions) remains $O(T \cdot \sum_{i=1}^{S} \mathsf{H}_k(X_i))$, but $\sum_{i=1}^{S} \mathsf{H}_k(X_i)$ is not generally equal to $\mathsf{H}_k(\mathbf{X})$, and so an optimal algorithm may be able to reduce the bound due to the δ dependence allowed. By application of Lemma 4.2, an optimal algorithm's bound is $S_{opt}(\mathbf{X}, \delta) = T(1-\delta)(\sum_{i=1}^{S} \mathsf{H}_k(X_i)) + T \cdot O(\delta)$. We have the following theorem regarding the compressed size of the sensor streams.

Theorem 5.4. *Given a set* $\mathbf{X}$ *of sensor streams from an empirically* (δ, m)*-local sensor system, for any* $0 \leq \delta < 1 - \varepsilon$*, where* δ *may vary based on the sensor system and* $\varepsilon > 0$ *is an absolute constant,*

$$\mathrm{Enc}_{opt}(\mathbf{X}) = O\left(\max\left\{\delta T, S_{opt}(\mathbf{X})\right\}\right) \textit{ bits}.$$

Proof. As in the proof of Theorem 5.2, an optimal algorithm would compress each partition to take the greatest advantage of the dependence between clusters. It would achieve a space bound of

$$S_{opt}(\mathbf{X}, \delta) = T(1-\delta)\left[\sum_{i=1}^{S} \mathsf{H}_k(X_i)\right] - T \cdot O(\delta)$$

for each partition. The *PartitionCompress* algorithm compresses each partition to space

$$S_{opt}(\mathbf{X}) = T \sum_{i=1}^{S} \mathsf{H}_k(X_i).$$

The ratio is

$$\rho = \frac{S_{opt}(\mathbf{X})}{S_{opt}(\mathbf{X}, \delta)} = \frac{\sum_{i=1}^{S} \mathsf{H}_k(X_i)}{(1-\delta)[\sum_{i=1}^{S} \mathsf{H}_k(X_i)] - O(\delta)}.$$

The rest of the proof proceeds following the proof of Theorem 5.2 □

We are also interested in the LZ78 algorithm, since the dictionary created in the process of compression is useful for searching compressed text without uncompressing it [17]. While Kosaraju and Manzini [26] show that LZ78 does not achieve the optimal bound of $T \cdot \mathsf{H}_k(X)$, they show that it uses space at most $T \cdot \mathsf{H}_k(X) + O((T \log\log T)/\log T)$. In our context, this means that each cluster uses space $T \cdot \mathsf{H}_k(X) + O((T \log\log T)/\log T)$.

Theorem 5.5. *Given a set* $\mathbf{X} = \{X_1, X_2, \ldots, X_S\}$ *of sensor streams taken over a sufficiently long time* T *from an empirically* (δ, m)*-local sensor system, for any* $0 \leq \delta < 1 - \varepsilon$*, where* δ *may vary based on the sensor system and* $\varepsilon > 0$ *is an absolute constant,*

$$\begin{aligned}\mathrm{Enc}(\mathbf{X}) &= cT \sum_{i=1}^{S} \left(\mathsf{H}_k(X_i) + O\left(\frac{\log\log T}{\log T}\right)\right) \\ &= O\left(\max\left\{\delta T, S_{opt}(X, \delta), \frac{T \log\log T}{\log T}\right\}\right) \textit{ bits}.\end{aligned}$$

Proof. An optimal algorithm would compress each partition to take the most advantage of the dependence between clusters. It would achieve a space bound of

$$T(1-\delta)\left[\sum_{i=1}^{S} \mathsf{H}_k(X_i)\right] - T \cdot O(\delta).$$

In contrast, using LZ78 as the basis for *PartitionCompress* compresses each partition to

$$T \sum_{i=1}^{S} \mathsf{H}_k(X_i) + \sum_{i=1}^{S} O\left((T \log\log T)/\log T\right)$$

where S is the total number of clusters over all partitions. The ratio is

$$\begin{aligned}&\frac{\sum_{i=1}^{S} \mathsf{H}_k(X_i) + \sum_{i=1}^{S} O((\log\log T)/\log T)}{(1-\delta)[\sum_{i=1}^{S} \mathsf{H}_k(X_i)] - O(\delta)} \\ &= \frac{1}{1-\delta}\left[\frac{\sum_{i=1}^{S} \mathsf{H}_k(X_i) + \sum_{i=1}^{S} O((\log\log T)/\log T)}{[\sum_{i=1}^{S} \mathsf{H}_k(X_i)] - O(\delta)}\right].\end{aligned}$$

Here we consider the two possible cases for the relationship of $O(\delta)$ to $\sum_{i=1}^{S} \mathsf{H}_k(X_i)$:

1. *Case* $O(\delta) \geq \sum_{i=1}^{S} \mathsf{H}_k(X_i)$*:*

$$\frac{1}{1-\delta}\left[\frac{\sum_{i=1}^{S} \mathsf{H}_k(X_i) + \sum_{i=1}^{S} O((\log\log T)/\log T)}{[\sum_{i=1}^{S} \mathsf{H}_k(X_i)] - O(\delta)}\right]$$
$$\leq \frac{1}{1-\delta}\left[O(\delta) + \frac{\sum_{i=1}^{S} O((\log\log T)/\log T)}{O(\delta)}\right].$$

Choose T large enough so that $O((\log\log T)/\log T) < O(\delta)$. Then the ratio is

$$\leq \left[\frac{1}{1-\delta}\right] O(\delta) = O(\delta)$$

for a single one of the c partitions, or $O(\delta)$ total times $S_{opt}(\mathbf{X}, \delta)$, which is $O(\delta T)$ since $O(\delta) \geq \sum_{i=1}^{S} \mathsf{H}_k(X_i)$.

2. *Case* $O(\delta) < \sum_{i=1}^{S} \mathsf{H}_k(X_i)$*:*

$$\frac{1}{1-\delta}\left[\frac{\sum_{i=1}^{S} \mathsf{H}_k(X_i) + \sum_{i=1}^{S} O((\log\log T)/\log T)}{[\sum_{i=1}^{S} \mathsf{H}_k(X_i)] - O(\delta)}\right]$$
$$= \frac{1}{1-\delta}\left[O(1) + \frac{\sum_{i=1}^{S} O((\log\log T)/\log T)}{O(\sum_{i=1}^{S} \mathsf{H}_k(X_i))}\right].$$

Here, we consider two sub-cases based on the relationship between $O((S\log\log T)/\log T)$ and $\sum_{i=1}^{S} \mathsf{H}_k(X_i)$.

(a) *Case* $O((\log\log T)/\log T) \geq \sum_{i=1}^{S} \mathsf{H}_k(X_i)$*:*
Then the ratio is at most $O((1+\delta)(\log\log T)/\log T)$, for a total space of $O((1+\delta)((\log\log T)/\log T) S_{opt}(\mathbf{X}))$, which is $O(T(\log\log T)/\log T)$ since $O((\log\log T)/\log T) \geq \sum_{i=1}^{S} \mathsf{H}_k(X_i) > O(\delta)$.

(b) *Case* $O((\log\log T)/\log T) < \sum_{i=1}^{S} \mathsf{H}_k(X_i)$*:*
Then the ratio is at most

$$O\left(\frac{1}{1-\delta}\right) = O(1+\delta)$$

for a single one of the c partitions, or $O(1+\delta)$ total times $S_{opt}(\mathbf{X}, \delta)$, which is $O(S_{opt}(\mathbf{X}))$ since $\delta < 1 - \varepsilon$.

The final bound is $O(\max\{\delta T, S_{opt}(X, \delta), T(\log\log T)/\log T\})$ total space. □

We have now established $\mathrm{Enc}_{\mathrm{LZ78}}(\mathbf{X})$ in both statistical and empirical settings (Theorems 5.2 and 5.5 respectively) and shown that we achieve a space bound that is on the order of the optimal bound.

6. Conclusions

We introduced a sensor-based framework for kinetic data which can handle unrestricted point motion and only relies on past information. We analyzed our framework's encoding size and gave a c-approximation algorithm for compressing point motion as recorded by our framework for a constant c. Open questions include solving global statistical questions on kinetic data using this framework, e.g., answering clustering questions, finding the centerpoint, or finding the point set diameter. These solutions would likely be based on a range searching structure (developed as a result of the initial version of this paper) that operates without decompressing the data and is built within this framework [17].

In addition, we gave analyses of the compression algorithm in terms of both empirical and statistical entropy and considered a more realistic version of the framework that allows a limited version of independence between sensor observation streams. We showed that within all of these settings, the compression algorithm encoded the data to bounds on the order of the optimal. To extend these real-world considerations, and given that the framework was stated assuming a central processing node with global knowledge, it would be interesting to modify these algorithms to operate in a more distributed manner.

Acknowledgements

The authors thank anonymous reviewers for their helpful and detailed comments.

Appendix A. Technical lemmas

Lemma 3.1. *Consider two sensor streams X and Y over the same time period. Let $X + Y$ denote the componentwise sum of these streams. Then $\mathcal{H}(X+Y) \le \mathcal{H}(X,Y) \le \mathcal{H}(X) + \mathcal{H}(Y)$.*

Proof. To prove the first inequality, let $Z = X + Y$, and observe that $p(z) = \sum_{x+y=z} p(x,y)$. Clearly, if $x + y = z$, then $p(x,y) \le p(z)$. Thus,

$$\begin{aligned}\mathcal{H}(X+Y) &= -\sum_z p(z)\log p(z) \le -\sum_z \sum_{\substack{x,y\\ x+y=z}} p(x,y)\log p(x,y)\\ &= -\sum_{x,y} p(x,y)\log p(x,y) = \mathcal{H}(X,Y).\end{aligned}$$

By basic properties of conditional entropy (see, e.g., [10]), we have

$$\mathcal{H}(X,Y) = \mathcal{H}(X) + \mathcal{H}(Y|X) \le \mathcal{H}(X) + \mathcal{H}(Y),$$

which establishes the second inequality. □

Lemma 3.2. *Consider two strings $X, Y \in \Sigma^T$. Let $X + Y$ denote the componentwise sum of these strings.*

(i) *If X and Y are empirically independent, $\mathsf{H}_k(X,Y) = \mathsf{H}_k(X) + \mathsf{H}_k(Y)$.*
(ii) $\mathsf{H}_k(X,Y) = \mathsf{H}_k(X) + \mathsf{H}_k(Y|X)$.
(iii) $\mathsf{H}_k(X,Y) \le \mathsf{H}_k(X) + \mathsf{H}_k(Y)$.
(iv) $\mathsf{H}_k(X+Y) \le \mathsf{H}_k(X) + \mathsf{H}_k(Y)$.

Proof. We will not prove (i) here, since it will follow as a special case of Lemma 4.2 (by setting $\delta = 0$). To prove (ii), observe that by manipulation of the definitions

$$\begin{aligned}\mathsf{H}_k(X,Y) &= -\frac{1}{T}\sum_{x,y\in\Sigma^k} c(x,y)\left[\sum_{a,b\in\Sigma} \mathsf{p}_{X,Y}(xa,yb|x,y)\log \mathsf{p}_{X,Y}(xa,yb|x,y)\right]\\
&= -\frac{1}{T}\sum_{x,y\in\Sigma^k} c(x,y)\left[\sum_{a,b\in\Sigma} \mathsf{p}_{X,Y}(xa,yb|x,y)\log\big(\mathsf{p}_X(xa|x)\cdot \mathsf{p}_{X,Y}(yb|xa)\big)\right]\\
&= -\frac{1}{T}\sum_{x,y\in\Sigma^k} c(x,y)\left[\sum_{a,b\in\Sigma} \mathsf{p}_{X,Y}(xa,yb|x,y)\log \mathsf{p}_X(xa|x)\right]\\
&\quad -\frac{1}{T}\sum_{x,y\in\Sigma^k} c(x,y)\left[\sum_{a,b\in\Sigma} \mathsf{p}_{X,Y}(xa,yb|x,y)\log \mathsf{p}_{X,Y}(yb|xa)\right]\\
&= \mathsf{H}_k(X) + \mathsf{H}_k(Y|X).\end{aligned}$$

Symmetrically, we have $\mathsf{H}_k(X,Y) = \mathsf{H}_k(Y) + \mathsf{H}_k(X|Y)$.

To prove (iii), we first define the *empirical mutual information* of two strings X and Y, denoted $I_k(X;Y)$. Mutual information is a measure of the information that two strings share, or the amount by which the necessary encoding space of one is reduced by knowledge of the other.

$$I_k(X;Y) = \frac{1}{T}\sum_{x,y\in\Sigma^k} c(x,y)\sum_{a,b\in\Sigma} \mathsf{p}_{X,Y}(xa,yb|x,y)\log\frac{\mathsf{p}_{X,Y}(xa,yb|x,y)}{\mathsf{p}_X(xa|x)\mathsf{p}_Y(yb|y)}.$$

Now observe that since $\mathsf{p}_{X,Y}(xa|yb) = \mathsf{p}_{X,Y}(xa,yb|x,y)/\mathsf{p}_Y(yb|y)$,

$$\begin{aligned}I_k(X;Y) &= \frac{1}{T}\sum_{x,y\in\Sigma^k} c(x,y)\sum_{a,b\in\Sigma} \mathsf{p}_{X,Y}(xa,yb|x,y)\log\frac{\mathsf{p}_{X,Y}(xa|yb)}{\mathsf{p}_X(xa|x)}\\
&= -\frac{1}{T}\sum_{x,y\in\Sigma^k} c(x,y)\sum_{a,b\in\Sigma} \mathsf{p}_{X,Y}(xa,yb|x,y)\log \mathsf{p}_X(xa|x)\\
&\quad +\frac{1}{T}\sum_{x,y\in\Sigma^k} c(x,y)\sum_{a,b\in\Sigma} \mathsf{p}_{X,Y}(xa,yb|x,y)\log \mathsf{p}_{X,Y}(xa|yb)\end{aligned}$$

$$= -\frac{1}{T}\sum_{x,y\in\Sigma^k} c(x)\sum_{a,b\in\Sigma} \mathsf{p}_X(xa|x)\log \mathsf{p}_X(xa|x)$$
$$-\left(-\frac{1}{T}\sum_{x,y\in\Sigma^k} c(x,y)\sum_{a,b\in\Sigma} \mathsf{p}_{X,Y}(xa,yb|x,y)\log \mathsf{p}_{X,Y}(xa|yb)\right)$$
$$= \mathsf{H}_k(X) - \mathsf{H}_k(X|Y).$$

Symmetrically, we have $I_k(X;Y) = \mathsf{H}_k(Y) - \mathsf{H}_k(Y|X)$. By (ii) we have $I_k(X;Y) = \mathsf{H}_k(X) + \mathsf{H}_k(Y) - \mathsf{H}_k(X,Y)$. Since $I_k(X;Y)$ is clearly nonnegative, this implies that $\mathsf{H}_k(X,Y) \le \mathsf{H}_k(X) + \mathsf{H}_k(Y)$.

To prove (iv), let $Z = X + Y$. By the definition of empirical entropy we have

$$\mathsf{H}_k(X+Y) = -\frac{1}{T}\sum_{z\in\Sigma^k}\sum_{\substack{x,y\\ x+y=z}} c(x+y)\left[\sum_{g\in\Sigma}\sum_{\substack{a,b\\ a+b=g}} \mathsf{p}_Z\big((x+y)(a+b)\big|x+y\big)\log \mathsf{p}_Z\big((x+y)(a+b)\big|x+y\big)\right],$$

where $x+y$ is an outcome of length k and $a+b$ is an outcome of length 1 in the new string $X+Y$. By the same reasoning as in Lemma 3.1, $\mathsf{p}_{X,Y}(x,y) \le \mathsf{p}_Z(x+y)$. Substituting this relationship into our equation and, since we desire an upper bound, considering only cases in which

$$-\mathsf{p}_Z\big((x+y)(a+b)\big|x+y\big)\log \mathsf{p}_Z\big((x+y)(a+b)\big|x+y\big) \le -\mathsf{p}_{X,Y}(xa,yb|x,y)\log\big(\mathsf{p}_{X,Y}(xa,yb|x,y)\big),$$

we find that

$$\mathsf{H}_k(X+Y) \le -\frac{1}{T}\sum_{x+y\in\Sigma^k} c(x,y)\left[\sum_{a,b\in\Sigma} \mathsf{p}_{X,Y}(xa,yb|x,y)\log \mathsf{p}_{X,Y}(xa,yb|x,y)\right]$$
$$= \mathsf{H}_k(X,Y).$$

By (iii) we have $\mathsf{H}_k(X,Y) \le \mathsf{H}_k(X) + \mathsf{H}_k(Y)$, which implies that $\mathsf{H}_k(X+Y) \le \mathsf{H}_k(X) + \mathsf{H}_k(Y)$, as desired. □

References

[1] B. Adams, M. Pauly, R. Keiser, L.J. Guibas, Adaptively sampled particle fluids, ACM Trans. Graph. 26 (3) (2007).
[2] P.K. Agarwal, L.J. Guibas, H. Edelsbrunner, J. Erickson, M. Isard, S. Har-Peled, J. Hershberger, C. Jensen, L. Kavraki, P. Koehl, M. Lin, D. Manocha, D. Metaxas, B. Mirtich, D.M. Mount, S. Muthukrishnan, D. Pai, E. Sacks, J. Snoeyink, S. Suri, O. Wolefson, Algorithmic issues in modeling motion, ACM Comput. Surv. 34 (December 2002) 550–572.
[3] M.J. Atallah, Some dynamic computational geometry problems, Comput. Math. Appl. 11 (12) (1985) 1171–1181.
[4] B. Babcock, C. Olston, Distributed top-k monitoring, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, 2003, pp. 28–39.
[5] J. Basch, L.J. Guibas, Data structures for mobile data, J. Algorithms 31 (1) (April 1999) 1–28.
[6] K. Buchin, Algorithms for movement ecology. Presented at the Workshop on Geometric Computing Challenges, Rio de Janeiro, Brazil, June 2013.
[7] K. Buchin, T. Arseneau, S. Sijben, E.P. Willems, Detecting movement patterns using brownian bridges, in: Proceedings of the 20th International Conference on Advances in Geographic Information Systems, 2012, pp. 119–128.
[8] G. Cormode, S. Muthukrishnan, K. Yi, Algorithms for distributed functional monitoring, ACM Trans. Algorithms 7 (2) (2011).
[9] G. Cormode, S. Muthukrishnan, W. Zhuang, Conquering the divide: continuous clustering of distributed data streams, in: Proceedings of the IEEE 23rd International Conference on Data Engineering, 2007, pp. 1036–1045.
[10] T.M. Cover, J.A. Thomas, Elements of Information Theory, second edition, Wiley–IEEE, 2006.
[11] M. de Berg, M. Roeloffzen, B. Speckmann, Kinetic convex hulls, Delaunay triangulations and connectivity structures in the black-box model, J. Comput. Geom. 3 (1) (2012) 222–249.
[12] A. Deligiannakis, Y. Kotidis, N. Roussopoulos, Processing approximate aggregate queries in wireless sensor networks, Inf. Syst. 31 (8) (2006) 770–792.
[13] A. Deligiannakis, Y. Kotidis, N. Roussopoulos, Dissemination of compressed historical information in sensor networks, VLDB J. 16 (4) (2007) 439–461.
[14] P. Erdös, L. Lovász, Problems and results on 3-chromatic hypergraphs and some related questions, in: A. Hajnal, L. Lovász, V. Sos (Eds.), Infinite and Finite Sets, vol. 10, 1975, pp. 609–627.
[15] P. Ferragina, G. Manzini, Opportunistic data structures with applications, in: Proceedings of the 41st Annual Symposium on Foundations of Computer Science, 2000, pp. 390–398.
[16] S.A. Friedler, D.M. Mount, Compressing kinetic data from sensor networks, in: Proceedings of the Fifth International Workshop on Algorithmic Aspects of Wireless Sensor Networks, AlgoSensors, 2009, pp. 191–202.
[17] S.A. Friedler, D.M. Mount, Spatio-temporal range searching over compressed kinetic sensor data, in: Proceedings of the European Symposium on Algorithms, ESA, 2010, pp. 386–397.
[18] S. Gandhi, R. Kumar, S. Suri, Target counting under minimal sensing: complexity and approximations, in: Proceedings of the Workshop on Algorithmic Aspects of Wireless Sensor Networks, AlgoSensors, 2008, pp. 30–42.
[19] S. Gandhi, S. Nath, S. Suri, GAMPS: Compressing multi sensor data by grouping and amplitude scaling, in: Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, 2009.
[20] L. Guibas, Kinetic data structures, in: D. Mehta, S. Sahni (Eds.), Handbook of Data Structures and Applications, Chapman and Hall/CRC, 2004, pp. 23-1–23-18.
[21] L.J. Guibas, Sensing, tracking and reasoning with relations, IEEE Signal Process. Mag. 19 (2) (Mar 2002).
[22] A. Guitton, N. Trigoni, S. Helmer, Fault-tolerant compression algorithms for delay-sensitive sensor networks with unreliable links, in: Proceedings of the 4th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS), 2008, pp. 190–203.

[23] P. Gupta, R. Janardan, M. Smid, Fast algorithms for collision and proximity problems involving moving geometric objects, Comput. Geom., Theory Appl. 6 (1996) 371–391.
[24] D.A. Huffman, A method for the construction of minimum-redundancy codes, Proc. IRE 40 (9) (1952) 1098–1101.
[25] S. Kahan, A model for data in motion, in: Proceedings of the 23rd ACM Symposium on Theory of Computing, STOC'91, 1991, pp. 265–277.
[26] R.S. Kosaraju, G. Manzini, Compression of low entropy strings with Lempel–Ziv algorithms, SIAM J. Comput. 29 (3) (1999) 893–911.
[27] R. Krauthgamer, J.R. Lee, Navigating nets: simple algorithms for proximity search, in: Proceedings of the Fifteenth Annual ACM–SIAM Symposium on Discrete Algorithms, 2004.
[28] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, J. Anderson, Wireless sensor networks for habitat monitoring, in: ACM International Workshop on Wireless Sensor Networks and Applications, 2002, pp. 88–97.
[29] M.I.T. Media, Lab. The owl project, http://owlproject.media.mit.edu/.
[30] J.J. Monaghan, Smoothed particle hydrodynamics, Rep. Prog. Phys. 68 (2005) 1703–1759.
[31] D.M. Mount, N.S. Netanyahu, C. Piatko, R. Silverman, A.Y. Wu, A computational framework for incremental motion, in: Proceedings of the Twentieth Annual Symposium on Computational Geometry, 2004, pp. 200–209.
[32] S. Nikoletseas, P.G. Spirakis, Efficient sensor network design for continuous monitoring of moving objects, Theor. Comput. Sci. 402 (1) (2008) 56–66.
[33] J. Rissanen, Generalized Kraft inequality and arithmetic coding, IBM J. Res. Dev. 20 (3) (1976) 198–203.
[34] C.M. Sadler, M. Martonosi, Data compression algorithms for energy-constrained devices in delay tolerant networks, in: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems, November 2006, pp. 265–278.
[35] N. Saunier, T. Sayed, Automated analysis of road safety with video data, in: Transportation Research Record, 2007, pp. 57–64.
[36] E. Schomer, C. Theil, Efficient collision detection for moving polyhedra, in: Proceedings of the Eleventh Annual Symposium on Computational Geometry, 1995, pp. 51–60.
[37] E. Schomer, C. Theil, Subquadratic algorithms for the general collision detection problem, in: Abstracts 12th European Workshop Computational Geometry, 1996, pp. 95–101.
[38] C.E. Shannon, A mathematical theory of communication, Bell Syst. Tech. J. 27 (623–656) (July, October 1948) 379–423.
[39] B.J.M. Stutchbury, S.A. Tarof, T. Done, E. Gow, P.M. Kramer, J. Tautin, J.W. Fox, V. Afanasyev, Tracking long-distance songbird migration by using geolocators, Science (February 2009) 896.
[40] M. Treiber, Dynamic traffic simulation, http://www.traffic-simulation.de/, Jan 2010.
[41] M. Treiber, A. Hennecke, D. Helbing, Congested traffic states in empirical observations and microscopic simulations, Phys. Rev. E, Stat. Nonlinear Soft Matter Phys. 62 (2000) 1805–1824.
[42] C.R. Wren, Y.A. Ivanov, D. Leigh, J. Westbues, The MERL motion detector dataset: 2007 workshop on massive datasets, Technical report TR2007-069, Mitsubishi Electric Research Laboratories, Cambridge, MA, USA, August 2007.
[43] K. Yi, Q. Zhang, Multidimensional online tracking, ACM Trans. Algorithms 8 (2) (2012).
[44] J. Ziv, A. Lempel, A universal algorithm for sequential data compression, IEEE Trans. Inf. Theory IT-23(3) (May 1977).
[45] J. Ziv, A. Lempel, Compression of individual sequences via variable-rate coding, IEEE Trans. Inf. Theory 24 (5) (1978) 530–536.